AN ASHTON-TATE SOFTWARE SUPPORT PUBLICATION SEPTEMBER 1986

SEP 18 1986

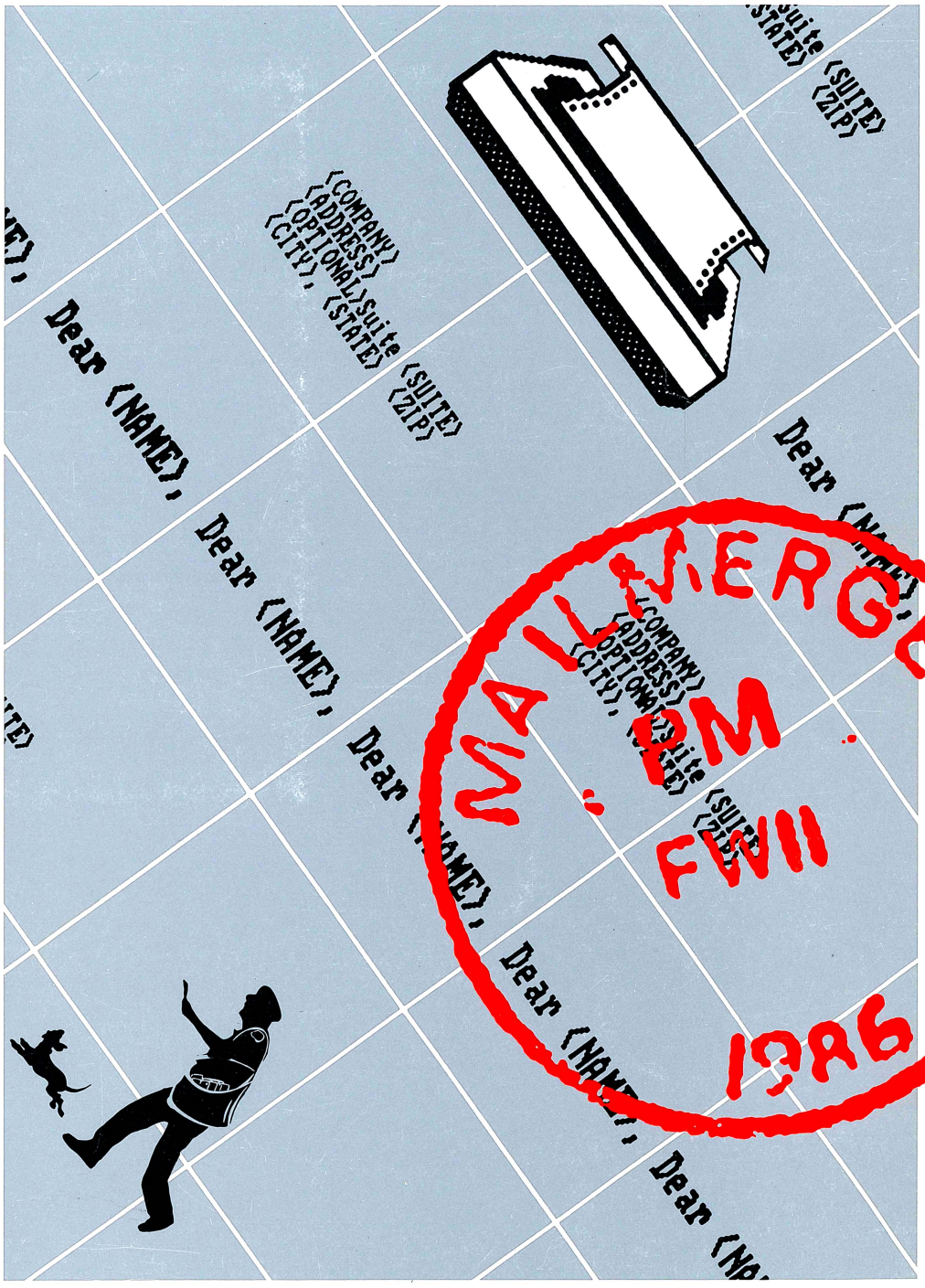
TechNotes

Mailmerge Application Programming

Sharing Data with MultiMate

Export to DIF in Framework II

dBASE III PLUS Change Summary —Part Three



TechNotes

TABLE OF CONTENTS

FEATURES

3 Mailmerge Application Programming
by Karen Robinson and Olivier Biggerstaff

12 Exporting to a VisiCalc DIF File
by Quinn Wildman and Karen Robinson

19 dBASE III PLUS Change Summary—Part Three
by Joseph Perret

26 Simulating the @...Say Command in FRED
by Quinn Wildman

31 Memo File Size Reduction
by Jack Hirsch

DEPARTMENTS

1 Read Me First
by Laurie Miller

35 Usage Tips

41 Anomalies

49 On-Line Services

54 MultiMate Technical Support

61 International Support

63 Letters

READ ME FIRST

by Laurie Miller

Our featured topic for this month's ***TechNotes*** is mailmerge, part of our ongoing series on reporting. Three articles cover this topic: Karen Robinson's and Olivier Biggerstaff's "Mailmerge Application Programming" for our dBASE III PLUS users, and two articles in the **MultiMate Technical Support** section. "Mailmerge Application Programming" presents four program modules that work together to perform a mailmerge in dBASE III PLUS. The system of programs takes data from fields in a database file and merges this into a prepared form letter. Also included are word-wrapping and error-checking modules.

In the **MultiMate Technical Support** section, we are pleased to bring you information on sharing data with MultiMate Advantage. "Sharing Data with MultiMate Advantage," by Liz Burke, our MultiMate Contributing Editor, and Karen Robinson, provides you with all of the information necessary to perform the merge. Another mailmerge-related article is "How to Sort and Print Info-Handler Datafiles," by Charles Gagliardi, one of MultiMate's documentation writers. This article discusses how you can use the Information Handling feature of MultiMate Advantage to more easily manipulate data used with the merge utility.

A useful program for Framework II users who also work with DIF files is discussed in "Exporting to DIF," by Karen Robinson and Quinn Wildman. Although Framework II has the capability of importing files in DIF format, it does not support an exporting routine. This program provides this capability as an add-in.

Also in this issue is part three of the dBASE III PLUS Change Summary. This portion of the change summary includes fixed anomalies that were reported for dBASE III in the February and March 1985 issues of ***TechNotes***.

For those of you who enjoy reading as many technical tidbits as you can get your hands on, turn to **Usage Tips**, **Anomalies**, **On-Line Services**, and **Letters**. We have heard your cry for more of this type of technical support; so, we are beefing up these monthly columns as much as we can. This month's **Usage Tips** column, for example, includes some neat "quickie" macros for Framework II, as well as many tips for dBASE III PLUS LAN users.

Last, but not least, you may have noticed that we have not published a cumulative index in this issue. We hope this won't be an inconvenience to you. The reason we dropped the index is that it is being completely redesigned. Many readers have commented via the ***TechNotes*** Reader Survey that the index is unusable and bulky. You just can't use it like you can a cross-referenced index to find the information you need. So, in response to your comments, we are currently redesigning and reorganizing the index so that it can be a useful and informative tool for you. We plan to publish the new and improved index in the December 1986 issue; this will be the Annual Cumulative Index. If you have any comments or suggestions about the ***TechNotes*** index (i.e., do you use it for the current or past issues, or both, or neither?), please write to me directly at the Ashton-Tate Software Support Center.

TechNotes

Publisher
Ashton-Tate Software Support Center
Jay Hanson, Director
Roy Moore, Manager

Editor-in-Chief
Laurie Miller

Editors
Christopher White

Technical Writer
Karen Robinson

Assistant Editor
Olivier Biggerstaff

Technical Staff
Larry Abame, Richard Avgelli, Stanley Ballenger, Rodney Barbati, Jeffrey Bersin, Robert Boies, Richard Bunter, Mike Carlisle, Ray Cortenbach, Fred Crownover, Michael Fiore, Ken Getz, Richard Goepel, Peter Gold, Stephen Goodman, Mina Gussin, Doug Haman, Scott Hanes, Mike Hedblom, Clint Helton, Jack Hirsch, David Howlett, Errol Jackson, Matt Johnson, Rosaline Keenan, Gail Kulikowsky, Steven Kurasch, Jerry Levy, Chris Lo Vasco, Tom Mangelsdorf, Jerry Mendes, Vincent Mendillo, Elizabeth Miranda, Jeffrey Moore, Keith Mund, Joseph Perret, Jaspal Rattan, John Regen, Deborah Ridolfo, Daryl Saybold, Steve Silverwood, Joseph Stoltz, Ralph Wakefield, Lisa Wanek, Quinn Wildman, Roger Wegehoft, Kari Wolff, Leonard Zerman

Contributing Editors
Paul Baukauskas (International), Liz Burke (MultiMate), Kent Irwin (On-Line Services)

Administrative Assistant
Kathy Smith

Illustrator
Stephen Goodman

Production Staff
D. A. Gray Bare, Tim Dunne, Jeff Elliott, Darrell Evers, Louis Hirsch, R. James Landers, Renée Maynard, Patty Pesek,

Product Development
Forefront Development Center: Marilyn McMahon, Greg Stikeleather
Glendale Development Center: Debby Moody

DISCLAIMER NOTICE

The information contained in this publication is designed to help you develop and improve your programming skills and to assist you in using Ashton-Tate products. Ashton-Tate assumes no responsibility whatsoever for the uses made of any of the software code in this issue, whether modified or unmodified.

If you discover what you believe is an anomaly (as defined under **Anomalies**) not presented in this issue or others, please send a detailed description of the possible anomaly to:

Ashton-Tate Software Support Center
20101 Hamilton Ave.
Torrance, CA 90502-1319

If the reported anomaly can be reproduced by our technicians, it may be added to a future issue of ***TechNotes***.

The policies that are described in ***TechNotes*** apply only to the United States (and possessions), Canada, and Mexico. Residents of other foreign countries should contact a local Ashton-Tate representative in their country for policy information and technical support.

Ashton-Tate and the authors will not be liable for special, incidental, consequential, indirect or other similar damages, even if we have been advised of the possibility of such damages. This means we are not responsible for damages or costs incurred as a result of loss of time, loss of data, loss of profits or revenues, or loss of use of the software. In addition, we are not responsible for damages or costs incurred in connection with obtaining substitute software, claims by others, inconvenience or similar costs.

No warranties, either expressed or implied, are made regarding the contents of this publication, its merchantability or fitness for any particular purpose, all of which are hereby expressly disclaimed. Note, however, that some states do not allow the exclusion or limitation of warranties or of direct, incidental or consequential damages, so these warranty limitations may not apply to you.

In no event will our liability for any damages to you or any other person ever exceed the price paid for your subscription to ***TechNotes***, regardless of any form of the claim.

Additional statements by agents, employees, distributors and dealers of Ashton-Tate do not constitute warranties by Ashton-Tate and do not bind Ashton-Tate.

TechNotes (ISSN 0883-6493) is an Ashton-Tate Software Support Center publication containing technical reference information on Ashton-Tate and MultiMate products.

Entire contents Copyright © Ashton-Tate 1986. All rights reserved. Commercial reproduction of the material appearing in ***TechNotes*** is forbidden without prior written permission. Direct all requests for reproduction permission to Ashton-Tate Software Support Center, 20101 Hamilton Ave., Torrance, CA 90502-1319 ATTN: Laurie Miller.

TechNotes is published monthly at a subscription price of $50.00 for one year. Additional rates: Canada and Mexico, $65.00; other countries, $75.00. Monthly issues are available in back issues for $10.00 each (1985 issues) and $7.00 each (1986 issues). ***TechNotes*** is published by the Ashton-Tate Software Support Center, 20101 Hamilton Ave., Torrance, CA 90502-1319. POSTMASTER: Send subscription inquiries and address changes to ***TechNotes*** Subscription Service, P.O. Box 3729, Escondido, CA 92025-0929.

dBASE II, dBASE III PLUS, and Framework II are registered trademarks of Ashton-Tate. dBASE, RunTime, FRED, and ASHTON-TATE are trademarks of Ashton-Tate. Some of the trademarks of other companies mentioned in this issue appear for identification purposes only.

The Ashton-Tate Software Support Center's hours are 6:00 a.m. to 4:30 p.m., PST, Monday through Friday, except on Thursday when we close at 4:00 p.m. To reach our technicians for technical support, call (213) 329-0086. Be sure to have your software serial number ready for the technician.

Mailmerge Application Programming

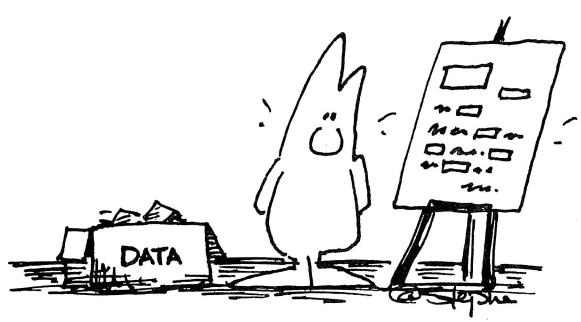
by Karen Robinson and Olivier Biggerstaff

This month, ***TechNotes*** expands its discussion on reporting with this article on merging data from a database file into a custom form letter. The program is generic and can, therefore, be used with any database file and any form letter. This article outlines the requirements for merging data from a database file of your choice, and gives a brief discussion of each routine that makes up the mailmerge program.

The main function of a mailmerge program is to bring data from a database file into a format resembling a typewritten letter. The mailmerge program presented here prints one record per page. Multi-paged letters are not supported. You embed fields within the letter, and the mailmerge program inserts the data into the letter, trimming trailing blanks and formatting the text according to the margins you specified. To use this program you need to create a form letter, and the four programs: Main.PRG, Merge.PRG, Wordwrap.PRG, and Checker.PRG.

The Form Letter

The form letter is a text file that contains the contents of the letter with the field names to merge from the database file. To create the letter, use MODIFY COMMAND or any text editor that creates a standard ASCII file. Type the letter in the format of your choice and embed the field names within the text. You can also include the dBASE III PLUS functions, DATE() and TIME() within the letter. Dates are printed in the format set by SET DATE and SET CENTURY, and TIME() is printed in its default format, hh:mm:ss. No other dBASE III PLUS commands or functions can be used within the form letter. If you do include any other dBASE III PLUS command or function, it is treated as printable text. Be sure to embed these supported functions in delimiters as you would any other variable text.



Use the backslash symbol (\) in column 1 of each line in a list of items, such as an address. The backslash symbol indicates to the program to print on a new line (refer to the Wordwrap section of this article for details). For example,

```
\John Doe
\123 Anywhere Dr.
\Culver City, CA 90230
```

results:

```
John Doe
123 Anywhere Dr.
Culver City, CA 90230
```

If the backslashes are left out, the data prints on one line:

```
John Doe 123 Anywhere Dr. Culver City, CA 90230
```

Field names must be delimited with either of the accepted field delimiters: angle < > or square brackets [ ]. Use the angle brackets to delimit field names that always print, such as the city or state in an address. Use the square brackets to indicate the field as optional; that is, the field may or may not have data in it. If an optional field contains no data, a blank line does not print. For example, if your database file has two address fields, and the second one is empty but marked as optional, the program does not print a blank line in place of the field:

```
\<Company>
\<Address1>
\[Address2]
\<City>, <State>  <Zip>
```

results in:

```
Ashton-Tate
20101 Hamilton Ave.
Torrance, CA 90250
```

However, if the field is empty and not marked as optional, a blank line prints:

```
\<Company>
\<Address1>
\<Address2>
\<City>, <State>  <Zip>
```

results in:

```
Ashton-Tate
20101 Hamilton Ave.

Torrance, CA 90250
```

These symbols, [ ] and < >, are reserved as field delimiters only, and cannot be used within the text of your letter.

If you want blank spaces to follow a field marked as optional, embed the blank spaces within the brackets instead of outside of the brackets:

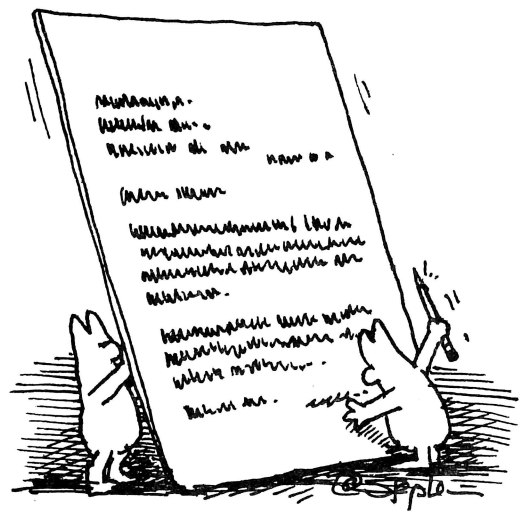
```
[Name   ]xxx instead of
[Name]  xxx
```

That way, if the optional field is empty, the embedded spaces are deleted, preventing extra spaces from printing in the text of the letter.

For fields that always print (delimited with angle brackets), place the blank spaces outside the delimiters:

< Name > xxx instead of
< Name > xxx

Extra spaces within the angle brackets are trapped as an error by the validation program (Checker.PRG).



The Main Program

Main.PRG is the starting point of the merge process. The program creates two database files: Strufile.DBF and Sysfile.DBF. Strufile.DBF is a structure extended file that contains the structure of your database file. It is used by the main program to create Sysfile.DBF, and by Checker.PRG to verify the correctness of the field names used in your form letter. Sysfile.DBF contains one field called Textdata, 254 characters wide; it stores the contents of your form letter, one record for each 254 characters of text. After these files are created, the validation program Checker.PRG is executed. If Checker.PRG detects a problem, it displays an error message and stops. If there are no problems, Main.PRG enters a DO WHILE loop to process one record at a time through Merge.PRG. The DO WHILE loop in this example processes all the records in the database file. If you want, you can use SET FILTER or another conditional statement to narrow the range of records to print. Once all the records are processed, the database files are closed; Strufile.DBF and Sysfile.DBF are ERASEd.

Merging Database Fields with Your Letter

Merge.PRG is the program that merges the data from the database file into the form letter. One record is processed at a time. It searches for the delimited field names in Sysfile.DBF and replaces them with the data from the current record of the database file. Once all the fields have been replaced, it then prints the form letter. The letter is actually printed through a word-wrap routine that wraps the text at the right column when it exceeds the specified line width. This process repeats for each record in the database file.

Word Wrapping Long Lines

As stated above, Wordwrap.PRG is a routine that prints the letter, word wrapping when necessary. Main.PRG passes two variables to Wordwrap.PRG: "offset" and "linelen." "Offset" stores the value of the left margin; its default value is 6. "Linelen" stores the value for line width; its default value is 65. You can change either of these default values in Main.PRG.

The word-wrap routine starts printing on a new line when it detects two consecutive **Returns**, or a backslash in column 1 of a line. Paragraphs are usually separated by a blank line, which means a **Return** follows the last line of the previous paragraph, and a **Return** precedes the blank line. If your paragraphs are not separated by a blank line, use the backslash symbol at the beginning of each paragraph. For example, the backslashes are not needed in this situation:

> This is paragraph one. <**Return**>
> <**Return**>
> This is paragraph two. <**Return**>
> <**Return**>

although, they are needed in this one:

> \This is paragraph one. <**Return**>
> \This is paragraph two. <**Return**>

Checking your Letter for Errors

Checker.PRG verifies that the field names cited in your form letter are spelled and delimited correctly. Field names with unbalanced delimiters, such as <Name or Name], are trapped as errors. Checker.PRG also notifies you if there are too many delimiters, such as << Name >> . If it does find an error, an error message displays and execution stops.

Setup

To set up the program, create a procedure file that contains Merge.PRG, Wordwrap.PRG, and Checker.PRG. Then create Main.PRG and your form letter. Replace the filenames printed in italics with your own filenames. To begin the program, type DO Main from within dBASE III PLUS.

```
* Program ...: Main.PRG
* Author ....: Olivier Biggerstaff
* Date .......: September 1, 1986
* Version ...: dBASE III PLUS or the Developer's Release
* Note(s) ...: Main mailmerge program
*
SET TALK OFF
SET PROCEDURE TO Mergproc
SET SAFETY OFF
* ---Define the left margin and the line length.
* ---The maximum line length is 254-offset.
offset = 6
linelen = 65
* ---Build necessary system files. Specify your own filenames
* ---for <YourDatabase> and <YourLetter>.
SELECT 1
USE <YourDatabase>
COPY STRUCTURE EXTENDED TO Strufile
* ---Strufile.DBF contains the structure of the database.
```

```
SELECT 3
USE Strufile
* ---Temp.DBF is to contain the structure for Sysfile.DBF.
COPY TO Temp
SELECT 2
USE Temp
ZAP
APPEND BLANK
* ---Prepare to create a database for the template.
REPLACE Field_name WITH "textdata",;
        Field_type WITH "C",;
        Field_len WITH 254
USE
CREATE Sysfile FROM Temp
ERASE Temp.DBF
USE Sysfile
APPEND FROM <YourLetter>.TXT SDF
* ---Sysfile.DBF contains your form letter.
GO TOP
* ---Validate the delimiters and field names.
* ---Stop processing if error is .T. from Checker.PRG.
error = .F.
* ---Running Checker.PRG is optional.
DO Checker
IF .NOT. error
   SELECT 2
   USE sysfile
   * ---Print one letter for each record in the database.
   SELECT 1
   * ---Define your filter here.
   SET PRINT ON
   DO WHILE .NOT. EOF()
      DO Merge
      EJECT
      SKIP
   ENDDO
   SET PRINT OFF
   * ---An error was found in the form letter.
ELSE
   ? CHR(7)
   ? "Processing cancelled"
ENDIF
* ---Close all files and remove unwanted files.
CLEAR ALL
ERASE Strufile.DBF
ERASE Sysfile.DBF
RETURN
* EOP Main.PRG
```

```
PROCEDURE Checker
* Author ....: Olivier Biggerstaff
* Date ......: September 1, 1986
* Version ...: dBASE III PLUS or the Developer's Release
* Note(s) ...: Checks for inconsistencies in the form letter.
*              Called by Main.PRG.
*
SET EXACT ON
* ---Validate each line of the form letter.
DO WHILE .NOT. EOF() .AND. (.NOT. error)
```

```
recnum = RECNO()
data = TRIM(UPPER(textdata))
* ---Process each field of one line from the form letter.
DO WHILE LEN(data) > 0 .AND. .NOT. error
   * ---Check for duplicate delimiters.
   IF AT("<<",data) > 0 .OR.;
      AT(">>",data) > 0 .OR.;
      AT("[[",data) > 0 .OR.;
         AT("]]",data) > 0
         @ 24,0 SAY "Too many delimiters - Line: "+LTRIM(STR(recnum))
         error = .T.
      ELSE
         pos1 = AT("<",data)
         pos2 = AT(">",data)
         pos3 = AT("[",data)
         pos4 = AT("]",data)
         delimtype = "<"
         * ---Determine the type of delimiter used.
         IF pos3 > 0 .AND. (pos1 = 0 .OR. pos3 < pos1) .OR. pos4 > 0
            pos1 = pos3
            pos2 = pos4
            delimtype = "["
         ENDIF
         nodelim = .F.
         IF LEN(data) > pos1
            nodelim = AT(delimtype,;
                      SUBSTR(data,pos1+1,pos2-pos1-1)) > 0
         ENDIF
         * ---Check for a missing right delimiters.
         IF pos1 > 0 .OR. pos2 > 0
            IF pos2 = 0 .OR. nodelim
               @ 24,0 SAY "Missing right delimiter " +;
                          "- Line: " + LTRIM(STR(recnum))
               error = .T.
            ELSE
               * ---Check for a missing left delimiters.
               IF pos1 = 0 .OR. pos1 > pos2
                  @ 24,0 SAY "Missing left delimiter " +;
                             "- Line: " + LTRIM(STR(recnum))
                  error = .T.
               ELSE
                  * ---Extract the field name
                  * ---from the template line.
                  subdata = SUBSTR(data,pos1 + 1,pos2 - pos1 - 1)
                  IF pos2 < LEN(data)
                     data = SUBSTR(data,pos2+1)
                  ELSE
                     data = ""
                  ENDIF
                  * ---Determine whether or not the field exists.
                  * ---Bypass this step if a dBASE function is used.
                  IF subdata <> "DATE()" .AND. subdata <> "TIME()"
                     SELECT 3
                     LOCATE FOR field_name = subdata
                     IF (delimtype = "<" .AND.;
                        AT(" ",subdata) > 0) .OR. EOF()
                     IF LEN(subdata) > 10
                        subdata = SUBSTR(subdata,1,10)
                     ENDIF
                     @ 24,0 SAY "Field name not "+;
                                "recognized: '"+subdata+"'- Line: ''+;
                                LTRIM(STR(recnum))
                     error = .T.
```

```
                  ENDIF
               ENDIF
            ENDIF
            SELECT 2
         ENDIF
      ELSE
         data = ""
      ENDIF
   ENDIF
ENDDO
SKIP
ENDDO
SET EXACT OFF
SET SAFETY ON
* ---Return to Main.PRG.
RETURN
```

```
PROCEDURE Merge
* Author ....: Olivier Biggerstaff
* Date ......: September 1, 1986
* Version ...: dBASE III PLUS or the Developer's Release
* Note(s) ...: Merges one record of the database with the
*              text of the form letter. This program assumes
*              that all delimiters and field names are correctly
*              specified. Sysfile.DBF is created by Main.PRG.
*
SELECT 2
USE Sysfile
* ---Toprint contains the text to be printed.
toprint = ""
DO WHILE .NOT. EOF()
   optional = .F.
   * ---Extract one line from the form letter
   * ---and remove the backslash character if any.
   data = TRIM(IIF(SUBSTR(textdata,1,1)="\",SUBSTR(textdata,2),textdata))
   DO WHILE LEN(data) > 0
      * ---Determine if optional-field delimiters are used.
      optional = .F.
      pos1 = AT("<",data)
      pos2 = AT(">", data)
      pos3 = AT("[",data)
      IF pos3 > 0 .AND. (pos1 = 0 .OR. pos3 < pos1)
         pos1 = pos3
         pos2 = AT("]",data)
         optional = .T.
      ENDIF
      * ---Extract the field name from the form letter.
      IF pos1 > 0 .AND. pos2 > pos1
         subdata = SUBSTR(data,pos1 + 1,pos2 - pos1 - 1)
         * ---Keep a count of the optional spaces after the field name.
         optspace = AT(" ",subdata)
         IF optspace > 0
            optspace = SPACE(LEN(subdata) - optspace + 1)
         ELSE
            optspace = ""
         ENDIF
         * ---Determine the field type in the database
         * ---and convert the contents to character.
         SELECT 1
```

```
         fieldtype = TYPE(TRIM(UPPER(subdata)))
         DO CASE
         CASE fieldtype = "C"
            subdata = TRIM(&subdata)
         CASE fieldtype = "N"
            subdata = LTRIM(STR(&subdata))
         CASE fieldtype = "D" .OR. UPPER(subdata) = "DATE()"
            subdata = DTOC(&subdata)
         CASE fieldtype = "L"
            subdata = IIF(&subdata,"Yes","No")
         CASE UPPER(subdata) = "TIME()"
            subdata = TIME()
         ENDCASE
         IF LEN(subdata) = 0
            optspace = ''''
         ENDIF
         * ---Build the string to print by adding
         * ---the contents of data to toprint.
         toprint = toprint + SUBSTR(data,1,pos1 - 1) + subdata + optspace
         SELECT 2
         IF pos2 < LEN(data)
            data = SUBSTR(data,pos2 + 1)
         ELSE
            data = ""
         ENDIF
      ELSE
          toprint = toprint + data
          data = ''
      ENDIF
   ENDDO
   * ---Print one block of text with the wordwrap routine.
   DO Wordwrap WITH "toprint", linelen
   * ---Determine if the next line is a new line.
   newline = IIF(LEN(TRIM(textdata)) = 0,.T.,.F.)
   SKIP
   IF SUBSTR(textdata,1,1) = "\" .OR.;
      LEN(TRIM(textdata)) = 0 .OR.;
      newline
      * ---Print any remaining text before starting a new line.
      IF .NOT. optional .OR. LEN(TRIM(toprint)) > 0
         ? TRIM(SPACE(offset) + toprint)
      ENDIF
      toprint = ""
   ENDIF
ENDDO
* ---Print any remaining text before ending.
? TRIM(SPACE(offset) + toprint)
SELECT 1
RETURN
* EOP Merge.PRG
```

```
PROCEDURE Wordwrap
* Author ....: Olivier Biggerstaff
* Date ......: September 1, 1986
* Version ...: dBASE III PLUS or the Developer's Release
* Note(s) ...: Prints a block of text of a specified line length,
*             word-wrapping, leaving the remaining text in the
*             variable pointed to by strname.
*
```

```
PARAMETERS strname, length
string = &strname
DO WHILE LEN(string) >= length
   * ---Look for the first space at the end of the line.
   counter = length
   DO WHILE counter > 0 .AND. SUBSTR(string,counter,1) <> ""
   counter = counter - 1
   ENDDO
   * ---If a space was not found, print anyway.
   counter = IIF(counter = 0,length,counter)
   ? SPACE(offset) + SUBSTR(string,1,counter - 1)
   string = LTRIM(SUBSTR(string,counter))
ENDDO
* ---The remaining text goes back into the text variable.
&strname = TRIM(string) + " "
RETURN
* EOP Wordwrap.PRG
```


SOFTWARE SUPPORT TECHNICIANS

The Coasts Are Clear

ASHTON-TATE is clearing the path in the competitive microcomputer software marketplace with our renowned combination of sophisticated software packages and qualified personnel. We currently have opportunities on both coasts for Software Support Technicians, in our Los Angeles operation and at our MultiMate facility in Hartford, Connecticut.

MULTIMATE
East Hartford, Connecticut

Requirements include previous MultiMate™ word processing experience, support experience, training abilities, sharp analytical and troubleshooting skills and familiarity with DOS, hardware configurations and printer interfacing. Exposure to Local Area Networks a plus. Familiarity with MultiMate™, dBASE III PLUS™, and FRAMEWORK II™ required.

ASHTON-TATE
Torrance, California

Requirements include an indepth working knowledge of dBASE™ along with the problem solving skills needed to go on-line (telephone) to interface with ASHTON-TATE developers, dealers and end-users. Verbal and written communication skills are important, as is your proven ability to write and debug dBASE II™ programs. Familiarity with MS-DOS and CP/M is also required.

In return for your expertise and personal initiative, you can expect first class resources, generous benefits that include relocation expenses and a company you can grow with—not out of. For consideration, send your resume with salary history, in confidence to: **Mr. Tony Faison, ASHTON-TATE, 20101 Hamilton, Torrance, CA 90502.** An Equal Opportunity Employer.

ASHTON·TATE

Exporting to a VisiCalc DIF File

by Quinn Wildman and Karen Robinson

This article provides information on and a methodology to export Framework II spreadsheets to a VisiCalc format. As you may know, Framework II imports VisiCalc DIF files, but it does not export to them.

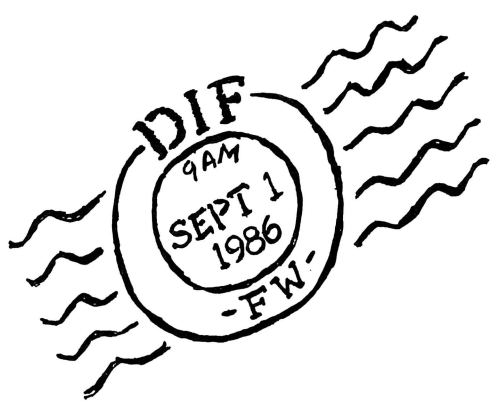
The program presented in this article, Fw_2_dif.FW2, exports a Framework II spreadsheet to the DIF file format described below. All other frame types must be copied into a spreadsheet frame before executing the program. Convert one spreadsheet frame at a time; if you have several spreadsheets in a containing frame, convert each one separately. All data types are stored as either character or numeric. Dates are represented as character strings and logical constants as numbers (1 for #TRUE, 0 for #FALSE). Numbers are stored with 10 decimal places. Formulas are not converted, but the results of the formulas are.

A DIF file is a standard ASCII text file with fixed length fields. Records (or rows) are called tuples, and fields (or columns) are called vectors.

There are two parts to the DIF file format: a header section and a data section. The header section contains a description of the file structure, and the data section contains the actual data. In the header, there are four standard and several optional elements. Fw_2_dif.FW2 writes only the standard elements to the header; all other options are ignored.

Header Section

The following is an example of a DIF file header:



```
TABLE 0,1'              <------ VisiCalc version 1
"Frame Name"            <------ Object name
TUPLES
0,Number of Rows        <------ Rows (records)
""
VECTORS
0,Number of Columns     <------ Columns (fields)
""
DATA 0,0
""                      <------ Data follows
```

The first element in the header, TABLE, contains the version number of the VisiCalc DIF file and the spreadsheet frame name. The second element, TUPLES, contains the number of rows in the spreadsheet. The third element, VECTORS, contains the number of columns in the spreadsheet. The fourth and last element, DATA, indicates that all elements to follow are data values and not header elements. (For detailed information on the structure of a DIF file, refer to Appendix A, "DIF Technical Specifications," in Donald H. Beil's *The DIF File*.)

Data Section

The data section consists of a series of tuples (spreadsheet rows) and an end-of-data marker, EOD, after the last tuple. Each tuple begins with a beginning-of-tuple marker, BOT, followed by a series of vectors (column information). Below is a diagram of this basic structure:

```
-1,0                    <------ Beginning of a new tuple
BOT
vector elements
.
.
-1,0                    <------ Beginning of a new tuple
BOT
vector elements
.
.
-1,0                    <------ End-of-data marker
EOD
```

Each vector is made up of two lines that contain the following information:

> Type Indicator, Number Value
> String Value

Type Indicator

The first item is the type indicator, followed by a comma and the number value. The type indicator specifies whether the vector is numeric, character, or special: 0 indicates numeric, 1 indicates character, and -1 indicates either the beginning of a tuple or the end of file.

Number Value

The number value depends on the value stored to the type indicator. If the type indicator is 0, then the number value is the actual numeric data item. If the type indicator is 1 or -1, then the number value is 0.

String Value

The string value further defines the number value. If the type indicator is 0 (data is numeric), then the number value represents one of four different data conditions with the string value storing these four conditions.

1. V means the number value is the actual numeric data.
2. NA means the data is not available, such as an empty cell or #N/A!.
3. ERROR indicates an invalid calculation or any Framework II error constant (except #N/A!).
4. TRUE means logical true (#TRUE) and FALSE means logical false (#FALSE).

If the type indicator is 1 (data type is character), then the string value contains the actual character string data.

How Data is Represented

Following are examples of how vector elements represent Framework II data types.

Character Strings (including dates):

```
1,0
<"Character String">
```

Numbers:

```
0,<Numeric Data>
V
```

#TRUE:

```
0,1
TRUE
```

#FALSE:

```
0,0
FALSE
```

Error Constants (except #N/A!):

```
0,0
ERROR
```

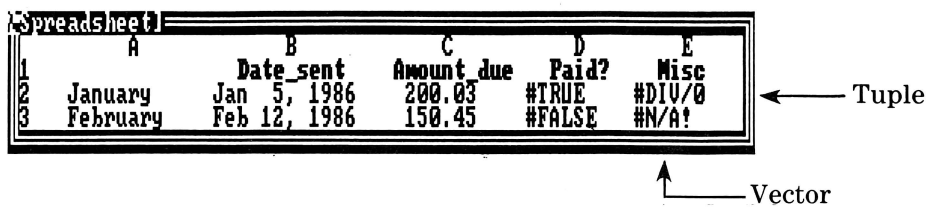
Empty Cell or #N/A!:

```
0,0
NA
```

The following chart summarizes the way in which the DIF file format defines Framework II data types:

Framework II Data Type	Type Indicator	Number Value	String Value
Character	1	0	Character Data
Dates	1	0	"MM/DD/YYYY"
Numeric	0	Numeric Data	V
Logicals			
#TRUE	0	1	TRUE
#FALSE	0	0	FALSE
Error constants			
#VALUE	0	0	ERROR
#REF	0	0	ERROR
#DIV/O	0	0	ERROR
#NUM	0	0	ERROR
#NULL	0	0	ERROR
#TBD	0	0	ERROR
#NAME?	0	0	ERROR
#NA!	0	0	NA
An empty cell	0	0	NA
Beginning of tuple	-1	0	BOT
End-of-data	-1	0	EOD

The example below illustrates what a DIF file looks like when exported through Framework II.



The resultant DIF file:

```
TABLE
0,1
"Spreadsheet"                    Spreadsheet name
TUPLES
0,3                              Three records (rows)
""
VECTORS
0,6                              Six fields (columns)
""
DATA                             Data elements follow
0,0
""
-1,0                             First row
BOT
0,0                              Column A, row 1 (empty cell)
NA
1,0                              Column B, row 1
"Date_sent"
1,0
"Amount_due"
1,0
```

```
"Paid?"
1,0
"Name"
1,0
"Misc"
-1,0                    <-------  Second row
BOT
1,0                     <-------  Column A, row 2
"January"
1,0                     <-------  Column B, row 2
"01/05/1986"
0,200.0300000000
V
0,1
TRUE
1,0
"John"
0,0
ERROR
-1,0                    <-------  Third row
BOT
1,0                     <-------  Column A, row 3
"Feburary"
1,0
"02/12/1986"
0,150.4500000000
V
0,0
FALSE
1,0
"David"
0,0
NA
-1,0                    <-------  End-of-data marker
EOD
```

Setting Up the Program

To set up the program, create an outline frame with two subframes. Label the containing frame Fw_2_dif and the two subframes Main and Datefunc. Type the respective programs, as indicated below, into the formula area of the frames. Then, through Setup, add this routine to the Disk:Export:Add-in submenu. For instructions on how to do this, refer to "Add-ins for Converting Other Formats," on page 10-19 of the *Using Framework II* manual.

Once the program is included as an Export Add-in, highlight the spreadsheet frame to export and execute the program from the Disk:Export submenu. The resultant file will have a .DIF extension.

```
; Fw_2_dif.FW2
; Framework II to DIF Exporter
;
;    by Quinn Wildman
;       September 1, 1986
;
@if(@key <> {F5}, @iserr(@Fw2dif.main))
```

```
; Main.FW2
; Main routine to convert a spreadsheet to a DIF file.
;
@local(pos, row, width, height, top, bottom, frame,
       g, n, temp1, temp, bot, v, na, error, true,
       false, text2, eod
),
; Initialize text constants.
bot := "-1,0
BOT
",
v := "
V
",
na := "0,0
NA
",
error := "0,0
ERROR
",
true := "0,1
TRUE
",
false := "0,0
FALSE
",
text2 := "1,0
""&temp1&"""
",
eod := "-1,0
EOD
",
@echo(#OFF),
; Check current frame as spreadsheet.
@if(@frametype <> 6,
   @list(@echo(#ON),
         @prompt("Frame is not a Spreadsheet"),
         @return(0)
   )
),
frame := @setselection,
@performkeys("{Dnlevel}{Ctrl-Home}"),
top := @setselection,
@performkeys("{Ctrl-End}"),
; Calculate vectors.
width := @panel1,
; Calculate tuples.
height := @panel2,
bottom := @setselection,
@performkeys("{Uplevel}"),
@echo(#ON),
; Create header.
temp := "TABLE
0,1
""&frame&"""
TUPLES
0,"&@integer(height)&"
"""
VECTORS
```

```
0,"&@integer(width)&"
""""""
DATA
0,0
""""""
",
g     := "@get(" & top & ":" & bottom &")",
temp1 := @g,
row   := 1,
@eraseprompt,
; Covert row.
@while(temp1 <> #NULL!,
    pos := 0,
    @prompt("Processing row "&@integer(row)),

    ; Insert begin of tuple.
    temp := temp & bot,
    @while(pos < width,

        ; Check if cell is numeric.
        @if(@isnumeric(temp1),
           temp := temp & "0," & @decimal(temp1,10) & v,

           ; Check if cell is not available.
           @if(@isna(temp1),
               temp := temp & na,

               ; Check if cell is an error constant.
               @if(@iserr(temp1),
                   temp := temp & error,

                   ; Check if cell is logical.
                   @if(@and(@not(@isalpha(temp1)),@iserr(@date1(temp1))),
                       @if(temp1,
                           temp := temp & true,
                           temp := temp & false
                       ),
                       @list(

                            ; Check if cell is a date.
                            @if(@not(@iserr(@date1(temp1))),
                                temp1 := @datefunc(@date1(temp1))
                            ),
                            temp := temp & text2
                        )
                    )
                )
            )
        ),
        pos := pos + 1,
        n   := "@next(" & top & ":" & bottom & ")",
        g   := "@get(" & top & ":" & bottom & ")",
        @n,
        temp1 := @g
    ),
    pos := 0,
    row := row + 1,
),

; Put in end of data.
temp := temp & eod, ,
@echo(#OFF),
@performkeys("{Ctrl-C}Etempframe{Return}"),
```

```
; DIF information must be put in text file
; first in order to @writetextfile.
tempframe := temp,
@writetextfile(frame & ".DIF",tempframe),
@performkeys("{Del}"),
@echo(#ON),
@eraseprompt,
@prompt("Export to DIF now completed")
```

```
; Datefunc.FW2
; Routine to covert a string in date1 format to a string in the
; form mm/dd/yy.
;
; item1 ::= <string> in the form of date1
;
@local(year,month,day),
@if(@mid(@item1,1,3)="Jan",month:="01"),
@if(@mid(@item1,1,3)="Feb",month:="02"),
@if(@mid(@item1,1,3)="Mar",month:="03"),
@if(@mid(@item1,1,3)="Apr",month:="04"),
@if(@mid(@item1,1,3)="May",month:="05"),
@if(@mid(@item1,1,3)="Jun",month:="06"),
@if(@mid(@item1,1,3)="Jul",month:="07"),
@if(@mid(@item1,1,3)="Aug",month:="08"),
@if(@mid(@item1,1,3)="Sep",month:="09"),
@if(@mid(@item1,1,3)="Oct",month:="10"),
@if(@mid(@item1,1,3)="Nov",month:="11"),
@if(@mid(@item1,1,3)="Dec",month:="12"),
@if(@mid(@item1,6,1)=",",
@list(
     day  := "0" & @mid(@item1,5,1),
     year := @mid(@item1,8,4)
),
@list(
     day  := @mid(@item1,5,2),
     year := @mid(@item1,9,4)
  )
),
@return(month&"/"&day&"/"&year)
```

---

dBASE III PLUS Change Summary—Part Three

by Joseph Perret

This is the third part of the four-part change summary listing anomalies reported in ***TechNotes*** for dBASE III version 1.1 and the Developer's Release fixed in dBASE III PLUS. This part lists anomalies reported from the February and March 1985 issues of ***TechNotes***. As was the case last month, each entry is followed by an explanation of the change made to the command or function. Some anomalies have been repaired outright and now work as documented. Others, occurring when an internal limit was breached or incorrect syntax employed, now produce appropriate error messages.

Next month, we will publish the fourth and last part of this change summary for issues April 1985 through June 1985.

@...GET

Formerly, if a memory variable was initialized to zero, dBASE III would not allow input larger than 10 digits when using @...GET , even if the PICTURE clause was used. The problem would not occur if a field was used rather than a memory variable.

Now in dBASE III PLUS and the Developer's Release, @...GET accepts input as large as the PICTURE clause, regardless of the declared size of the memory variable.

Reported in the February 1985 issue of ***TechNotes***, page D3-3.

@...SAY

Formerly, the A and ! PICTURE clause functions were mutually exclusive. Essentially, you could not combine both to limit data entry to alphabetic characters and force the letters to uppercase. If A and ! were used together, only the second function would be in effect. For example:

PICTURE "@A!" was equivalent to PICTURE "@!"
PICTURE "@!A" was equivalent to PICTURE "@A"

Now, although the dBASE III PLUS documentation states that these functions cannot be used together, in fact, they work together properly.

Reported in the February 1985 issue of ***TechNotes***, page D3-11.

@...SAY

Formerly, the zero-suppression function of the PICTURE clause, @Z, would not suppress the decimal point when it was used with a noninteger value of zero.

Now in dBASE III PLUS and the Developer's Release, the @Z function suppresses the decimal point for a value of zero.

Reported in the March 1985 issue of ***TechNotes***, page D3-7.

@...SAY...GET

Formerly, if you issued an @...SAY...GET that placed the first character of the GET in column 80, the computer would freeze, requiring a warm boot. If the column coordinate was changed to 79, no error message displayed and the GET wrapped to the next line on the screen.

Now in dBASE III PLUS, any @...SAY...GET that accesses coordinates off the screen produces the error message "Position is off the screen."

Reported in the March 1985 issue of ***TechNotes***, page D3-7.

COPY TO

Formerly, any form of COPY [STRUCTURE] TO <filename> ignored the status of SET SAFETY if the command was issued in a command file overwriting the target database file without warning.

Now in dBASE III PLUS and the Developer's Release, COPY honors the status of SET SAFETY, regardless of whether COPY is issued from a command file or from the dot prompt.

Reported in the March 1985 issue of ***TechNotes***, page D3-8.

COPY TO DELIMITED

Formerly, using COPY TO <filename> DELIMITED WITH without a delimiter was not trapped as an error and caused one of two problems.

1. If there was a space after the WITH, the error message "Unrecognized phrase/keyword in command" displayed.
2. If there was no space, the database file was COPYed but the target file contained all blank records.

Now in the Developer's Release and dBASE III PLUS, an incomplete COPY TO <filename> DELIMITED command returns the error message "Syntax error."

Reported in the March 1985 issue of ***TechNotes***, page D3-8.

CREATE

Formerly, if you pressed **Return** in the first field of the CREATE screen and then resumed editing by pressing any key other than **Return**, the structure of the database file saved would be corrupted. A subsequent LIST of the STRUCTURE might freeze the computer.

Now in the Developer's Release and dBASE III PLUS, a database file CREATEd in this way is not damaged.

Reported in the February and May 1985 issues of ***TechNotes***, page D3-12.

CTOD()/DTOC()

Formerly, CTOD() and DTOC() were not inverse functions in all cases. The exception occurred when the date was blank and the character string was in the form, " / / ". To compare for a blank date, you had to use the DTOC() function rather than CTOD(). For example,

```
? blank_date = CTOD(" / / ") <------ Returned (.F.)
? DTOC(blank_date) = " / / " <------ Returned (.T.)
```

Now in dBASE III PLUS, CTOD() and DTOC() are truly inverse functions. You can compare for a blank date by using either function.

Reported in the February 1985 issue of ***TechNotes***, page D3-1.

DO WHILE

Formerly, if an extra ENDDO was added to a command file, an infinite loop would result. This occurred under dBASE III version 1.0 only. In version 1.1, the extraneous ENDDO did not interfere with normal execution, but did not return an error message.

Now in the Developer's Release and dBASE III PLUS, an extra ENDDO in a command file produces the error message "Mismatched DO WHILE and ENDDO."

Reported in the February 1985 issue of ***TechNotes***, page D3-12.

Exponentiation Display

Formerly, when raising an expression to a power in dBASE III, an overflow condition would occur if both the exponent and the expression were literal numbers greater than or equal to 10. For example,

```
? 10^10
**********
```

You could work around this problem by STOREing either the expression or the exponent to a memory variable. For example,

```
num = 10
? 10^num
10000000000.00
```

Now in the Developer's Release and dBASE III PLUS, large numbers are stored in scientific notation. The Developer's Release puts all numbers in this format, regardless of size. dBASE III PLUS puts numbers in scientific notation up to 10^98. Larger numbers are displayed as an overflow with asterisks.

Reported in the March 1985 issue of ***TechNotes***, page D3-9.

PARAMETERS

1. Formerly, misspelling PARAMETERS or omitting the command when passing PARAMETERS produced an error message not relevant to the actual error. The error message appeared as follows,

```
Syntax error
?
-> eturn
```

Now in the Developer's Release and dBASE III PLUS, this produces the error message "No PARAMETER statement found."

Reported in the February 1985 issue of ***TechNotes***, page D3-13.

2. Formerly, when you executed a DO with a field in the PARAMETER list, dBASE III would give the message, "Variable not found." To use a field name in the parameter list, it was necessary to specify the alias. For example, the following worked,

```
USE Filea
DO <filename> WITH Filea->Field1
```

but the following did not.

```
USE Filea
DO <filename> WITH Field1
```

Now in the Developer's Release and dBASE III PLUS, you can pass a field as a PARAMETER without using the alias, if it is in the current work area.

Reported in the February 1985 issue of ***TechNotes***, page D3-3.

PCOL()/SET MARGIN TO

Formerly, PCOL() would not return the current printer head position when used in conjunction with SET MARGIN TO. Instead, it returned the current print position plus the margin offset. For example,

```
STORE "Hello" TO var1
STORE "there" TO var2
SET DEVICE TO PRINT
@ 10,10 SAY var1
@ PROW(), PCOL() + 1 SAY var2
SET MARGIN TO 10
@ 11,0 SAY var1
@ PROW(), PCOL() + 1 SAY var2
RETURN
```

printed the following:

```
Hello there
Hello there
```

Now in the Developer's Release and dBASE III PLUS, PCOL() properly returns the current printer head position.

Reported in the February 1985 issue of ***TechNotes***, page D3-13.

REPORT FORM

1. Formerly, the REPORT FORM printed memo fields longer than one page continuously with no page breaks.

   Now in dBASE III PLUS, memo fields longer than one page print with proper page breaks.

   Reported in the February 1985 issue of ***TechNotes***, page D3-13.

2. Formerly, when NOEJECT was used and the report did not fill the last page, the line counter was not reset to zero. Therefore, when the next report was printed, the first page would print an incorrect number of lines.

   Now in dBASE III PLUS, NOEJECT does not interfere with subsequent REPORT FORMs.

   Reported in the February 1985 issue of ***TechNotes***, page D3-13.

3. Formerly, REPORT FORMs with two levels of subtotals that used the "Eject after subtotals" option formatted the first page of the report differently from all other pages. The difference was slight, but the first page showed the subgroup heading on the line directly below the major group. All other pages had a blank line between the subgroup and the major group headings.

   Now, the Developer's Release and dBASE III PLUS put a blank line between the subgroup and major group headings on every page.

   Reported in the February 1985 issue of ***TechNotes***, page D3-13.

4. Formerly, when CREATEing a REPORT with subtotals, answering yes to the "Summary report only? (Y/N)" option could cause two problems. The REPORT FORM would either print only the contents of the fields being totaled, or it would freeze the computer, requiring a cold boot.

   Now in the Developer's Release and dBASE III PLUS, the "Summary report" option executes correctly.

   Reported in the February 1985 issue of ***TechNotes***, page D3-13.

5. Formerly, there was a problem with printing multiple line headings. Sometimes when printing a combination of character and numeric information, the second and third lines of the column headings were shifted to the left.

   Now in the Developer's Release and dBASE III PLUS, multi-line headings that combine numeric and character information print properly.

   Reported in the March 1985 issue of ***TechNotes***, page D3-10.

SET

1. Formerly, the option to open **INDEX** files under the **Files** menu did not open the specified index file but also closed all active index files in the current work area.

   Now in the Developer's Release and dBASE III PLUS, the INDEX option properly opens the selected index file.

   Reported in the March 1985 issue of ***TechNotes***, page D3-11.

2. Under the Disk menu, the DISK DRIVE SEARCH PATH is used to set the internal dBASE III path. If this path has already been set in CONFIG.DB or by using the dBASE SET PATH command, the path name shows when this option is selected. Formerly, when either **Return** or **Esc** was pressed at the selection, the path name was erased. Therefore, if you selected the PATH option and did not wish to change it, you had to re-enter the path name.

   Now in the Developer's Release and dBASE III PLUS, the PATH option of the SET command is not altered unless explicitly changed.

   Reported in the March 1985 issue of ***TechNotes***, page D3-11.

SET ALTERNATE

Formerly, dBASE III did not send a CHR(10) to an alternate file. The following command file,

```
SET ALTERNATE TO x
SET ALTERNATE ON
?? "first LF"
?? CHR(10)
?? "second LF"
SET ALTERNATE OFF
CLOSE ALTERNATE
```

generated the following file with no linefeed,

```
first LFsecond LF
```

Now in the Developer's Release and dBASE III PLUS, a (CHR(10)) can be sent to an alternate file.

Reported in the February 1985 issue of ***TechNotes***, page D3-4.

SET FUNCTION

Formerly, the maximum number of characters (including semicolons) that could be programmed into a function key was 30. Exceeding this limitation was not trapped as an error. Instead, the extra characters wrapped to the next function key. This problem also caused incorrect display of the function key values in full-screen SET.

Now in the Developer's Release and dBASE III PLUS, the programmable function keys may be SET TO any string up to the 254-character command line limit.

Reported in the March 1985 issue of ***TechNotes***, page D3-11.

SORT

Formerly, attempting to SORT with a FOR condition based on a single date or a range of dates gave the error message "00% Sorted *** STACK OVERFLOW ***" and you were dropped out of dBASE III to the operating system if the DTOC( ) was compared to a literal date string. For example,

```
SORT ON <exp> TO <filename> FOR DTOC(date) = "99/99/99"
```

If the literal date string was blank, the SORT appeared to execute normally, but no records were written to the target database file.

Now in the Developer's Release and dBASE III PLUS, SORT fully supports FOR conditions that contain date comparisons with DTOC( ) and blank dates.

Reported in the March 1985 issue of ***TechNotes***, page D3-12.

Simulating the @...SAY command in FRED

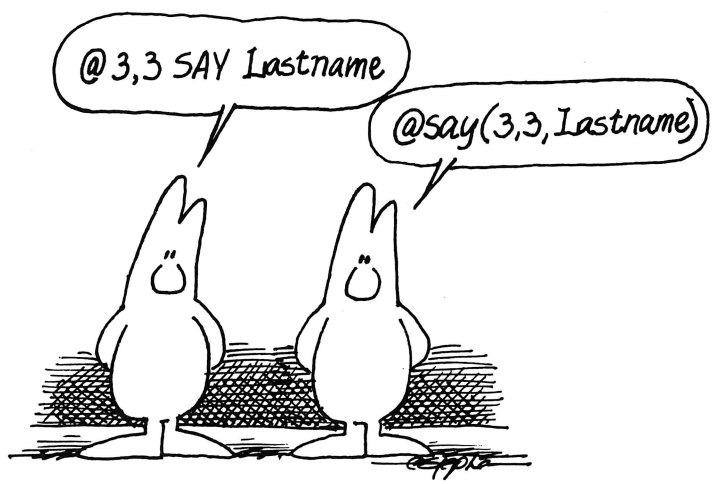
by Quinn Wildman

Generally, mailmerge is used to create customized reports in Framework II. For more flexibility, you can simulate the @...SAY command from dBASE to place values at row and column postions programatically as you would in a typical dBASE program.

The following program, Say, simulates the dBASE @..SAY command allowing you to produce custom formatted reports in just this way. It works by generating an outline with a word frame for each new page. If you have a five-page report, the resulting outline would have five subframes and look like this:

Report
I. Page 1
II. Page 2
III. Page 3
IV. Page 4
V. Page 5

By writing output to frames prior to printing allows you to add any of the Framework II formatting options before you print your report. An additional advantage to using this approach is that you save your report to a file, something you can't do with @...SAY in dBASE III or dBASE III PLUS.



Before you run your report, first call the initialization program, Say.Init. This program first checks for an empty word frame called report on the desktop. If it does not exist, Say.Init creates it. The frames lines per page, the top margin, and the bottom margin settings are then read to determine the number of lines of text per page. Each frame created in the outline has a maximum print area of < lines per page > - < top margin > - < bottom margin > lines of text in it. Additionally, each frame ends with a begin-new-page marker. If you want to use the defaults of the Print:Format:Options menu, you need to call Say.Init. If you want your report to have other settings, first create an empty word frame, give it the parameters that you wish your report to have, and then call Say.Init.

In dBASE, the @...SAY command has three critical parameters: row, column, and display expression. Say works similarly. Call it as follows, @Say(< row > , < column > , < expression >), passing the row and column parameters as numeric values. Just as in dBASE, whenever the row value gets smaller, the display expression will be put on the next page. Unlike dBASE III, however, the row value can exceed 254. In fact, it can become as large as you wish. The row and column parameters must be greater then zero since row and column zero do not exist in a Framework II word frame. Call @Say each time you wish to have a new expression put into the report.

Here is how to create this program.

1. Create an empty word frame and name it Say.
2. **Dnlevel** and create five subframes called Main, Init, Page, Length, and Row.
3. Now, enter the following formulas in corresponding frames by pressing **EDIT FORMULA (F2)** and **ZOOM (F9)**. Then enter the formula. When you are finished press **Esc**. Note that the Page, Length, and Row frames do not have formulas.

```
; Say.FW2
;
; Main for driver printing to specified row and column
; coordinates.
;
;    by Quinn Wildman
;      September 1, 1986
;
@Say.Main(@item1,@item2,@item3)
```

```
; Say.Main
;
; Called by Say.FW2
;
; length ::= global variable - length of page
; page   ::= global variable - current page number
;
; Define local variables.
@local(row, column, expression),
row        := @item1,
column     := @item2,
expression := @item3,
```

```
@echo(#OFF),
@setselection("Report"),

; If there is no data in expression or it is an error constant
; then report does not need to be done.
@if(@and(@not(@isna(expression)),@not(@iserr(expression))),
    @list(
        @local(expr),

        ; All data types must be checked for
        ; and made character type.
        @if(@isalpha(expression),
            expr := expression
        ),

         ; Check for numeric type.
        @if(@isnumeric(expression),
            expr := @integer(expression)
        ),

        ; Check for logical type.
        @if(@not(@iserr(@if(expression,#TRUE,#FALSE))),
            @if(expression,
                expr := "TRUE",
                expr := "FALSE"
            )
        ),

        ; Check for date type.
        @if(@not(@iserr(@date4(expression))),
            expr := @date4(expression)
        ),

        ; If typeover not ON, turn it ON.
        @if(@not(@sense(#ET)),
        @performkeys("{Ctrl-E}T")
        ),
        @performkeys("{Ctrl-Dnlevel}{Ctrl-End}"),

        ; Turn on echo for panel to work.
        @echo(#ON),

        ; First check row.
        @if(row < @panel2,
            @list(
                 page := page + 1,
                      @performkeys("{Ctrl-E}B{Uplevel}{Ctrl-C}Epage"&
                               @integer(page)&"{Return}{Dnlevel}")
             )
        ),
          @if(@or(@mod(row,length) < @panel2, row < @panel2),
              @list(
                   page := page + 1,
                   row  := 1,
                   ; Make sure there is one space after first "
                   @performkeys(" {Ctrl-E}B{Uplevel}{Ctrl-C}Epage"&
                                 @integer(page)&"{Return}{Dnlevel}")
              )
         ),
         @while(@and(@mod(row,length) <> @panel2,
                     @mod(row,length) <> 0
                ),
                     @performkeys("{Return}")
```

```
            ),

            ; Now check column.
            @if(@panel1 < = column,
                 @performkeys(@rept(" ",column - @panel1))
                 @while(@panel1 > column,
                        @performkeys("{Leftarrow}")
                 )
             ),
             @performkeys(expr)
        )
    ),

; Turn typeover OFF.
@performkeys("{Ctrl-E}T")
```

```
; Say.Init
;
; Called by your report program to set up and initialize the
; proper working environment for Say.Main.
;
;
@echo(#OFF),
@local(lines,tm,bm),

; If Report has not been created, then create it.
@if(@iserr(@setselection("Report")),
     @performkeys("{Ctrl-C}EReport{Return}"),
      @list(
            @setselection("Report"),
            @if(@frametype < > 8,
                @list(
                      @prompt("Report must be an empty word" &
                              "frame, please reconfigure"),
                      @return(0)
                 )
            )
      )
),
page := 1,

; Make left margin be 0 and create page 1.
@performkeys("{Ctrl-P}PFP00{Return}{Esc}{Dnlevel}{Ctrl-C}E"&
             "page1{Return}{Uplevel}"),

; Perform calculations determine page length.
@performkeys("{Ctrl-P}FPL{F6}{End}{Return}"),
lines := @value(@textselection),
@performkeys("{Return}HT{F6}{End}{Return}"),
tm := @value(@textselection),
@performkeys("{Return}FB{F6}{End}{Return}"),
bm := @value(@textselection),
@performkeys("{Esc}{Esc}"),
length := lines - tm - bm,
@echo(#ON)
```

The following example shows how to take information from a database frame and put it in a report using Say to format the output image. To create the program, create an empty word frame, enter the program formula into the formula area, and when you are done, press **Esc**. To execute, highlight your report frame and press **F5**.

```
; Calling.FW2
;
; Example report program that prints formatted output with Say.
;
;    by Quinn Wildman
;       September 1, 1986
;
; Assign the global variable an initial value.
Say.Row := 1,

; Initialize output frame and system variables
@Say.Init,

; Reset all regions of the database.
@reset(<your field reference>),

; Print all unfilterd records in the database.
@while(@get(<your key field reference>) <> #NULL!,

   ; Print your detail record. Always refer to your
   ; row co-ordinate as Say.Row.
   @Say(Say.Row, 1, @get(<your field reference>)),

   ; Advance to the next row.
   @next(<your field reference>),

   ; Advance the row pointer.
   Say.Row := Say.Row + 1
)
```

Once the report is generated, all you need to do is put special formatting in your report, such as headers and footers, and then print it. Unfortunately formatting such as bold, underline, and italics is not directly supported by Say. Either enter special styling attributes in the output frames or send codes to your printer with Say to get special styling. An example of how to send a code for an Epson printer to do compressed print is as follows,

```
@Say(row,column,@chr(15))
```

Memo File Size Reduction

by Jack Hirsch

As you may know, dBASE III PLUS memo files increase in size whenever you APPEND, EDIT, or DELETE memo field information. For this reason, it is a good idea to maintain your memo files by reducing their size periodically. In the April 1985 issue of ***TechNotes*** on page D3-5, there is a reference note that demonstrates the basic algorithm to recover unused space in the memo file. The commands in this note are simple and direct, but if you maintain a number of database files with memo fields, you may prefer to automate the procedure and adding an interface at the same time. Additionally, if you are creating an application, you may want to build the automated routine into your application's overall file maintenance module.

The following program, Memoslim.PRG, is written to meet both needs. The program starts by prompting you for the name of a database file and prompts you if the database file or its memo (.DBT) file is not found. If the specified files are found, Memoslim.PRG executes another program, Memotrim.PRG, that actually performs the commands to reduce the memo file's size. When the reduction process is complete, the total number of bytes recovered is displayed and you have the option to reduce the size of another memo file.

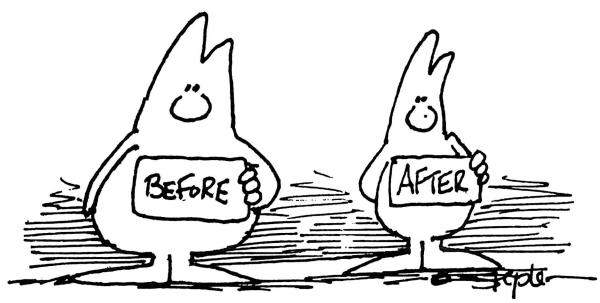
To set up, create the following programs, Memoslim.PRG and Memotrim.PRG. Then CREATE a database file called, Getfile.DBF with the following structure.

Field	Field Name	Type	Width	Dec
1	Line	Character	80	

To run, type DO Memoslim.

In addition to running Memoslim.PRG, you can run the reduction routine, Memotrim.PRG, as a stand-alone program or procedure. To run the Memotrim, use the following command line in your command files or from the dot prompt.

DO Memotrim WITH " < database filename > "

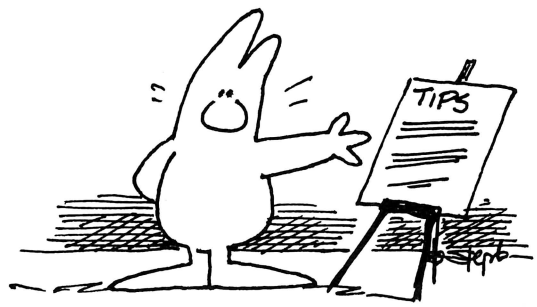


```
* Program..: Memoslim.PRG
* Author...: Jack Hirsch
* Date.....: September 1, 1986
* Version..: dBASE III PLUS
* Note(s)..: Program to reduce the size of memo files.
*
* ---Set environment.
SET TALK OFF
SET SAFETY OFF
SET BELL OFF
SET DECIMAL TO 0
SET STATUS OFF
* ---Declare local variables.
PRIVATE fixing, getfile, i, newsize, ok, oldname, oldsize
fixing = .T.
CLEAR
@ 01,00 SAY "Memo File Maintenance"
@ 02,00 TO 02,79
@ 22,00 TO 22,79
* ---Begin primary loop.
DO WHILE fixing
   oldname = SPACE(8)
   @ 5,00 CLEAR TO 19,79
   @ 16,00 TO 18,79
   @ 17,02 SAY "Enter the name of a database file " +;
               "or press [RETURN] to quit >"
   @ 17,63 SAY oldname + ".DBF"
   ok = .F.
   * ---Begin secondary loop.
   DO WHILE .NOT. ok
      oldname = SPACE(8)
      @ 17,63 GET oldname
              PICTURE "@!"
   READ
   oldname = TRIM(oldname)
   DO CASE
   CASE "" = oldname
      * ---If Return was pressed, exit the program.
      EXIT
   CASE .NOT. FILE("&oldname..DBF");
        .OR. (.NOT. FILE("&oldname..DBT"))
      * ---Either the database file or its memo file was not found.
      SET BELL ON
      ?? CHR(7)
      @ 23,11 SAY IIF(.NOT. FILE("&oldname..DBF"),;
              ".DBF file ", ".DBT file ") +;
              "not found, Press any key to continue..."
      WAIT "" TO null
      @ 23,00 CLEAR
      SET BELL OFF
   OTHERWISE
      @ 05,00 CLEAR TO 19,79
      @ 16,00 TO 18,79
      @ 17,02 SAY "Size reduction for " +;
                  UPPER(oldname) + ".DBT in progress..."
      * ---If the file was found, proceed.
      ok = .T.
      * ---Run DOS DIR command, redirect the resulting
      * ---output to Getfile.TXT
      RUN DIR &oldname..DBT > Getfile.TXT
      USE Getfile
```

```
      ZAP
      APPEND FROM Getfile SDF
      * ---Get rid of all records except the one from
      * ---which we can extract the file's original size.
      DELETE FOR Line = " "
      PACK
      GO TOP
      oldsize = VAL(SUBSTR(Line,13,12))
      * ---Call memo file "size reduction" routine
      DO Memotrim WITH oldname
      @ 05,00 CLEAR TO 19,79
      @ 16,00 TO 18,79
      @ 17,02 SAY "Final stage..."
      * ---Run DOS DIR command again, output to Getfile.TXT
      RUN DIR &oldname..DBT > Getfile.TXT
      USE Getfile
      ZAP
      APPEND FROM Getfile SDF
      * ---Get rid of all records except the one from
      * ---which we can extract the file's new size.
      DELETE FOR Line = " "
      PACK
      GO TOP
      newsize = VAL(SUBSTR(Line,13,12))
      @ 05,00 CLEAR TO 19,79
      @ 16,00 TO 19,79
      @ 17,02 SAY oldname + ".DBT was reduced by " +;
                  LTRIM(STR(oldsize - newsize)) + " bytes."
         @ 18,02 SAY "Do you want to reduce the size of " +;
                     "another memo file? [Y/N] ";
                 GET fixing;
                 PICTURE "Y"
         READ
      ENDCASE
   ENDDO
ENDDO
CLEAR
* ---Clean up.
SET SAFETY ON
SET BELL ON
SET TALK ON
CLEAR ALL
RETURN
* EOP Memoslim.PRG
```

```
* Program..: Memotrim.PRG
* Author ..: Jack Hirsch
* Note(s)..: Program/procedure for COPYing and RENAMEing of a
*            database file. It can be called from Memoslim.PRG or
*            run stand-alone.
*
PARAMETERS oldname
SET DELETED ON
USE &oldname
COPY TO Tempfile
CLOSE DATABASE
ERASE &oldname..DBF
ERASE &oldname..DBT
RENAME Tempfile.DBF TO &oldname..DBF
RENAME Tempfile.DBT TO &oldname..DBT
SET DELETED OFF
RETURN
* EOP Memotrim.PRG
```

USAGE TIPS



dBASE Administrator

by David Howlett

"File is in use by another"

If the message "File is in use by another" or "Record is in use by another" appears while loading dBASE Administrator, the directory containing the dBASE Administrator program has been set to read-only.

The files DBA.LD1 and DBA.OVL must be read-only files, but the directory must be read/write/create. Directory privileges are controlled by the NET SHARE command on PC-NET and by the SYSCON program on Novell.

APPEND

APPENDing to a database file that is read-only does not generate an error message, but no records are added. The input screen appears and you can enter data. When you press **Ctrl-W** to save and exit, however, no records have been added to the database file.

The work-around is to make the attribute of the database file read-write and use Protect instead, to control access.

Directing Output to a Spool Device

On the PC-NET, Novell, and 3Com networks, you can direct printing to a network spool device by using either a network or dBASE Administrator command. Execute one the following network commands at the DOS prompt prior to running the Administrator.

Network	Network Command
PC-NET	NET USE LPT1: \\ < server > \ < printer name >
Novell	SPOOL /P = < a number between 0 and 5 >
3Com 3Plus	3P LINK LPT1: \\ < server > \ < printer name >

After you set the device and load the Administrator, redirect to the spool device with the following command:

```
SET PRINTER TO LPT1
```

Printing goes to the network spooler since LPT1 was redirected previously using network commands.

If the network device is not set and you are in the Administrator, use the following command lines to redirect printing to the spool device:

Network	dBASE Administrator Command Line
PC-NET	SET PRINTER TO \\ < server > \ < printer name > = LPT1
Novell	SET PRINTER TO \\SPOOLER
3Com 3Plus	SET PRINTER TO \\ < server > \ < printer name > = LPT1

Use these commands to redirect inside the Administrator only if you have not already redirected using NET USE, SPOOL, or 3P LINK.

There is no dBASE Administrator command to cancel a device (such as LPT1) from being spooled to. To cancel a device, RUN the appropriate network command. The following summarizes the list of appropriate command lines.

Network	dBASE Administrator Command Line
PC-NET	RUN NET USE LPT1: /D
Novell	RUN ENDSPOOL
3Com 3Plus	RUN 3P UNLINK LPT1:

Nondedicated Server

You may not be able to open a maximum of 15 dBASE III PLUS files if you are using a Novell nondedicated server. When a nondedicated server is booted, a fixed amount of memory is allocated to the Novell operating system. For example,

```
ANET$OS 320
```

is a command that tells Novell to reserve 320K for the operating system. If there is heavy network usage, it is possible for this not to be enough memory to maintain a large number of open files.

Ashton-Tate does not support the use of a Novell nondedicated server and so the work-around is to use a dedicated server.

RUN on Novell

Some application programs not specifically designed to operate on a network leave files open if executed by the dBASE Administrator RUN command on Novell. For example, if you execute RUN WordStar and then edit a file named Myfile.PRG, when you exit WordStar, Myfile.PRG is still an open file.

When WordStar terminates, it does not execute an instruct to close files. Instead, it expects DOS to close files for it. The dBASE RUN command on Novell does not perform this automatic closing of files because of the EOJOFF Novell command.

It is important to note that there are other programs that do not explicitly close files. The Novell dedicated file server has a command called CONSOLE that shows any open files on the network. Also note that this affects memo field editing as well as the RUN command.

There is no work-around.

Updating Index Files

Open index files modified by placing an RLOCK() and then executing an EDIT or REPLACE will not be written to until the user UNLOCKs that record.

Framework II

by Chris Lo Vasco

Changing the Style Attributes of Frame Labels

As you are probably aware, changing the style attribute of a frame label requires you to enter the label area of the frame, select text, and make the appropriate choice from the Words menu. If you try to change the style attribute of the label by highlighting the frame and invoking an attribute from the Words menu, you will change the attribute of the frame contents instead.

If you work with frames frequently, it is useful to create macros in order to do this. You can create them the easy way by recording them with the **Macro/Abbreviation** option of the Create pull-down menu or the more difficult way by entering them manually into the formula area of the macro frame. Recording has one major drawback, however. Recorded macros cannot make use of Framework II's "repeat key feature," where macros that execute as formulas can. Repeating macro keys is a useful way to save keystrokes and automate tasks on an ad-hoc basis. In this instance, if you have a containing frame with a number of contiguous subframes whose labels you want to style, pressing **Ctrl-** < number > and the macro key styles all the labels of the set.

Since the repeat key feature is essential to productivity, it is recommended that you define styling macros as formulas. To set them up in this way, create a frame for each attribute in the macro section of your library. Name each with the appropriate name of the macro key in the form {Alt- < key > }. Then enter the following formulas into the formula area of the respective macro frames.

```
; Normal.
@echo(#OFF),
@performkeys(" {F6}{End}{Ctrl-W}N{Return}{Dnarrow}"),
@echo(#ON)
                └This is a space.
; Bold.
@echo(#OFF),
@performkeys(" {F6}{End}{Ctrl-W}B{Return}{Dnarrow}"),
@echo(#ON)
                └This is a space.
; Underline.
@echo(#OFF),
@performkeys(" {F6}{End}{Ctrl-W}U{Return}{Dnarrow}"),
@echo(#ON)
                └This is a space.
; Italic.
@echo(#OFF),
```

```
@performkeys(" {F6}{End}{Ctrl-W}I{Return}{Dnarrow}"),
@echo(#ON)
              ↑
              └ This is a space.
```

Display a Frame While Editing a Formula

How many times have you been in a spreadsheet editing formulas in **ZOOM (F9)** and forgotten the cell reference you need? You find that you must leave **ZOOM (F9)** to check the cell reference and then return to complete the formula. This is more mentally aggravating than anything else, and definitely makes you wish there were another way!

The solution is a fairly simple macro that can be kept in your Library. This macro allows you to be in **ZOOM (F9)** and display the desktop view on top of the formula. The following figure demonstrates what it typically looks like.

To install this macro, simply create an empty word frame in the Macros section of the Library. Label this frame with the **Alt** key name you prefer, and enter the following formula into the formula section of the frame:

```
@local(func, frame, pos),
frame := @setselection,
@if((pos := @scan(".",frame) - 1) > 0,
    frame := @mid(frame,1,pos)),
func := "@display(" & frame & ")",
@Func,
@eraseprompt,
@prompt("Press any key to continue...",32),
@nextkey,
@echo(#ON)
```

When this macro is executed, the current frame displays on top of the formula as it appears on the desktop. Pressing any key then clears the frame's display, leaving you at the same position in the formula. This macro works with any type of frame and so can be used editing any formula.

There is one reminder, however. Do not modify the macro formula to include the full path to the innermost frame of the frame reference. The frame reference should only point to the outermost frame. If the frame reference includes the full path, and the formula you are editing is a spreadsheet cell or a database field, Framework II may fall into an endless loop that can only be terminated by warm booting your computer. As it is, the formula above is safe for use.

Erase and Center Prompt Function

A common FRED programming construct is to erase the prompt area and then display a centered message. You probably find yourself hard coding this simple set of actions mostly because a user-defined function hardly seems worth it. The following user-defined function doesn't elevate this task to worthiness but certainly makes the act of displaying a centered prompt easier. This function is straightforward. Pass as the sole parameter, a string or expression that results in a string. When executed, the prompt area erases and the string appears centered in the prompt area. For example,

@promptcenter("**My message is ...**")

To set up, create a frame in the user-defined function section of the Library, label it Promptcenter, and enter one of the following formulas in the formula area. The first formula performs no error checking.

```
; Promptcenter.
@eraseprompt,
@prompt(@item1,(82-@len(@item1))/2)
```

The following formula performs error-checking, handling "No parameter passed," "Parameter passed is not string," and "Parameter passed is greater than 80 characters."

```
; Promptcenter.
@if(@not(@iserr(@len(@item1))),
@eraseprompt,
@prompt(@item1,@max(0,(80-@len(@item1))/2)+1))
```

To give your prompt an attribute from the Words pull-down menu such as bold or italics, style the parameter of @Promptcenter with the attribute if it is a literal string.

Writing a Frame as a Frame or Text File

The following macro writes a highlighted frame as either a text file or a frame file with a user-supplied extension. This is handy to use just before editing formulas or text to write a copy of the frame to disk as a backup. By using different file extensions, you can save several copies of a frame at different stages of development.

To create the macro, create an empty word frame in the Macros section of the Library. Label it with the name of the **Alt** key you want to invoke it with, and enter the following formula into the formula section of the frame.

```
; Create a backup file as text or frame.
;
; by Christopher Lo Vasco
;
@local(extension, function, file, type),
type := " ",
@while(@and(type <> "T",type <> "F"),
   @eraseprompt,
   @prompt("Enter T for text file or F for frame file",20),
   type := @keyname(@nextkey)
),
extension := @inputline(
             "Enter the file extension " &
             "(for example, BAK) " &
             "and press <Return>","BAK"),
file := @mid(@setselection,1,8) &
      @if(@iserr(@len(extension)),"",".” & extension),
@eraseprompt,
@prompt("Writing " & file,39 - (@len(file) + 9)/2),
@if(type = "T",
   function := "@writetextfile(""",
   function := "@writeframefile("""
),
function := function & file & """," & @setselection & ")",
@Function,
@eraseprompt,
@prompt(file & " Written",39 - (@len(file) + 9)/2)
```

The macro is quite simple. When prompted, enter the type of file to be written to and the file extension desired, and the macro creates the backup file.

ANOMALIES

DEFINITION OF ANOMALY

We use the word "anomaly" to mean any one of the following:

- *INCORRECT COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified.*
- *INCOMPLETE COMMAND (or function) IMPLEMENTATION. A command or function does not execute as specified in every use of the command or function.*
- *DATA OR LOGICAL BOUNDARY LIMITATION. A command or function cannot perform an operation at a specified data or logical boundary.*

  NOTE: Ashton-Tate reserves the right to withhold any further information that may reveal proprietary or trade secret information.

dBASE Administrator

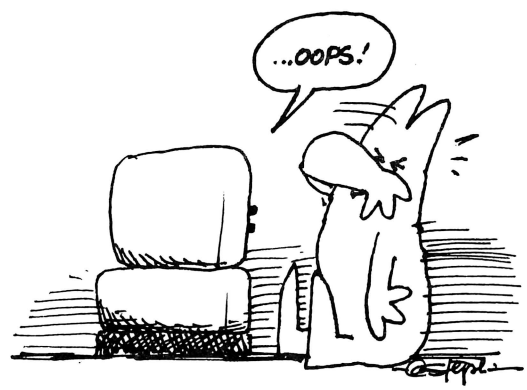
ON ERROR

ON ERROR does not execute correctly if TALK is ON and you encounter an error using LOCATE. The situation occurs when two workstations are using the same database file. For example, workstation one locks record five and then workstation two executes the following commands:

```
ON ERROR ? "This is an error"
SET TALK ON
LOCATE FOR RECNO() = 10
```

LOCATE encounters record five as locked and returns the error message, "Record is in use by another." The ON ERROR statement is not executed.

To work around, SET TALK OFF, and then ON ERROR executes as expected.



Protect

Protect sometimes creates encrypted .CRP files with embedded end-of-file characters. The original .DBF file, however, is not affected. At times, it appears that encrypting a 1,000-record database file creates a .CRP file with between one and four embedded end-of-file characters.

To work around, first COPY the STRUCTURE of the database file you want to encrypt. Then, in Protect, encrypt this new and empty database file. When you return to the Administrator, USE the empty .CRP file and then APPEND FROM the original database file containing your data. The result is the same as encrypting the original database file directly.

SET FILTER TO

The following sequence of commands places your workstation in an endless loop, requiring you to warm boot your computer.

```
SET EXCLUSIVE OFF
USE <database file>
SET FILTER TO <condition>
GO TOP
EDIT
```

The first EDIT screen appears and you can EDIT the record. Once, however, you reach the last record that matches the FILTER condition, a **PgDn** or any other key that advances to the next record places your workstation in an endless loop.

To work around, USE <database file> EXCLUSIVE or with an index file.

dBASE III PLUS

@...CLEAR TO

@...CLEAR TO does not erase a single line from the screen, although the @...TO draws a single line on the screen. The minimum vertical area @...CLEAR TO can erase is two lines. For example,

```
@ 2,0 TO 2,79
@ 2,5 CLEAR TO 2,25
```

draws a line from 2,0 to 2,79, but does not erase from 2,5 to 2,25.

To work around, use @ <row>,<col> with no parameter to erase the entire line. To erase portions, use @ <row>,<col> SPACE(<number of column positions>).

@...GET...READ

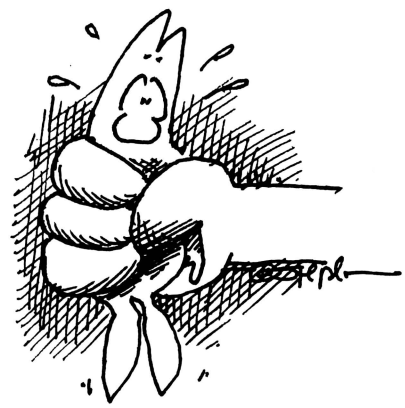
Performing a GET on a PRIVATE memory variable that you created in a lower program does not allow input when you RETURN to the calling program and issue a READ. For example,

```
DO WHILE .T.
    DO Test
    READ                    && Test.PRG sets up a PRIVATE variable.
ENDDO
* ---Test.PRG
mem1 = SPACE(2)
@ 12,41 GET mem1            && This GET will never be READ.
RETURN
```

Under certain circumstances, this causes the PICTURE template of some @...GET commands to be read into the GET as a literal value. This occurs if the command @...SAY DATE() is executed before READing the PRIVATE memory variable, and if the PICTURE template meets its own criteria as valid input. For example, a "9" is valid input to the PICTURE template "9." Once this error condition has occurred, every PICTURE template that meets its own criteria is READ as a literal value. READ ignores the GET and the value of the field or memory variable is not changed. The only way to stop this is to reload dBASE III PLUS. For example,

```
DO WHILE .T.
   mem = SPACE(6)
   @ 10,41 SAY DATE()
   @ 11,41 GET mem PICTURE "999999"
   READ
   DO Test
   READ              && Test.PRG sets up a PRIVATE variable.
ENDDO
```

In this example, the memory variable "mem" always READs as "999999" until you reload dBASE III PLUS.



ASSIST

1. Performing any full-screen edit on a record found with the Locate option of the Position menu disables the Continue option. When you return from a full-screen operation, Continue is no longer highlighted and so cannot be accessed.

   If you need to LOCATE and CONTINUE, specify the LOCATE in ASSIST and CONTINUE at the dot prompt. If you are EDITing FOR a condition, you can also EDIT with a FOR clause from the dot prompt. If you wish to remain in the ASSIST, CREATE a QUERY, and EDIT from this subset of records.

2. Whenever you choose a drive from any menu of available drives, that drive becomes the default.

CREATE REPORT

Pressing **Return** twice when highlighting the contents area of the Columns menu, creates an empty column in your REPORT FORM. Although you get no error message in CREATE REPORT, when you subsequently run the report, the error message "Syntax error in field expression" is returned.

To work around, MODIFY the REPORT and delete the empty column with **Ctrl-U.** You will find the location of the empty column in the Report Format area designated by a question mark.

CREATE SCREEN

1. If you are editing on the blackboard and you press **Ctrl-T** with the cursor on a line and a column to the left of a special character (!,?,.,@), the computer freezes requiring a warm boot.
2. If you terminate a box at column 79 and a row greater than 21 and then press **PgUp**, the box breaks into two sections, overwrites the menu line, and you cannot access line zero to edit it. A subsequent save of the screen and re-entering CREATE/MODIFY SCREEN does not correct the problem.

   Note that this problem occurs only when SET STATUS is ON. If you must terminate a box at this location, SET STATUS OFF before entering CREATE/MODIFY SCREEN.

FIND

You may receive the message "No find" when you use FIND with a macro (&) and in-line comments (&&) even though there are matching records in the index file.

FIND works correctly if the macro variable contains the entire key value, or if the key contains an embedded space and the variable contains key information up to the embedded space. For example, if the key field contains "FIRST" or "FIRST LAST," the commands

```
x = "FIRST"
FIND &x                 && This is a comment.
```

execute correctly. If, however, the variable contains "FIR," the example above returns the message "No find." Terminating the macro with a period does not change the result.

To include in-line comments with a FIND and a macro, the comment must begin flush against the macro or terminator. For example,

```
x = "FIR"
FIND &x                 && This does not work.
No find.
FIND &x.                && This does not work either.
No find.
FIND &x&&               This works.
FIND &x.&&              This works also.
```

INDEX

INDEX or REINDEX may produce the error message "File is not accessible" and damage index files in use. This occurs when the length of the key field and path to the .NDX file exceed 100 characters. This does not occur with small database files, but adding records eventually produces the error.

For example, if the key length is 85 characters,

INDEX ON < field > TO \Division\Three

returns the error message "File is not accessible," where

INDEX ON < field > TO \Division\Four

creates the index file as expected. The first path name is 15 characters and the second path name is 14. The length of the filenames and directory names are not important individually as long as the combined length of the path name does not exceed, in this example, 14 characters. Note that not all key expression and path name combinations exceeding 100 characters produce this error.

READ

dBASE III PLUS stops execution when you issue a READ command with no active GETs until you press a key. When this happens, the cursor is positioned to the right of its last screen position.

REINDEX

1. Occasionally, SET INDEX TO works, but a subsequent REINDEX returns the error message "File is not accessible." Other times, SET INDEX TO appears to work, but a GOTO returns the error message "Record is not in index."
2. On rare occasions, SET TALK has been shown to affect REINDEXing large database files returning the error message "File is not accessible." If you subsequently USE the database file with the index, the error message "Index damaged. REINDEX should be done before using data" is returned. If SET TALK is ON, however, REINDEX may work properly.

SET CATALOG TO

1. If you SET CATALOG TO < filename > when the maximum number of files is already open, dBASE III PLUS deletes all entries from the catalog, except those referring to files currently open.
2. If you SET PATH TO to the root directory of any drive, SET CATALOG TO < filename > returns the error message "File is not accessible," if it must create a catalog file on a directory other than the default directory. For example,

```
SET PATH TO B:\
* ---dBASE III PLUS is not on the root directory.
* ---C:\Test.CAT does not already exist.
SET CATALOG TO C:\Test
File is not accessible.<-------- Error message.
```

SET HISTORY TO

SETting HISTORY TO a number greater than the default of 20 interferes with DISPLAY HISTORY. The message "Press any key to continue..." appears, sometimes repeatedly, before the commands in HISTORY are displayed. The message also appears intermittently as HISTORY scrolls.

SET MESSAGE TO

SET MESSAGE TO accepts any length message, although only the first 79 characters display.

SET PRINT ON

There have been reported instances of the "Printer not ready" error message appearing on parallel as well as serial printers. This occurs when you use SET PRINT ON, SET DEVICE TO PRINT, TO PRINT, or SET PRINTER TO to output to the printer. Printing from DOS or other software packages works properly. There is no work-around if you experience this condition consistently.

SET PRINTER TO

1. SETting PRINTER TO a serial port works only intermittently. Any attempt to SET PRINTER TO COM1 or SET PRINTER TO COM2 may return the error message "Printer not ready" although the printer is on-line and ready to print.

   The work-around is to use the RUN command to load the resident portion of PC-DOS MODE into memory. For example,

   RUN MODE LPT1 = COM1

   This work-around may fail, however, if you are running under COMPAQ DOS.

2. Issuing SET PRINTER TO with no parameters stops all subsequent output to the printer. Commands such as LIST TO PRINT and REPORT FORM TO PRINT echo to the screen but do not print. In fact, while DISPLAY STATUS shows that output is assigned to the default port, LPT1, it has not been assigned to any device and therefore no printing can occur. A SET DEVICE TO PRINT may appear to freeze the computer, but you are still fully functional, except that no data is being output to the printer.

   To resume, SET PRINTER TO < device name > .

3. With a serial printer properly configured with MODE, if a SET PRINT ON command is issued, a later SET PRINT ON or SET DEVICE TO PRINT command returns the "Printer not ready" error message. SET PRINT OFF must be issued first. SET DEVICE TO PRINT followed by SET PRINT ON works as expected. This problem does not occur with a parallel printer.

SET UNIQUE ON

INDEXing with SET UNIQUE ON or INDEX ON UNIQUE works with small database files (less than 100 records), but fails in one of several ways when INDEXing larger database files.

1. INDEX with UNIQUE may seem to execute properly, but creates an index file that is neither unique nor complete.

2. INDEXing UNIQUE can freeze the computer causing one lost cluster on the disk. This leaves behind two files, an .NDX file containing 0 bytes and a larger .W44 file. The .W44 file is a temporary file created during the indexing process. Under normal circumstances, it is deleted when INDEX terminates.

3. INDEX with UNIQUE sometimes does not freeze the computer, but continues to create an index file until all the available disk storage is consumed, generating a "Disk is full" error message. If an INDEX ON UNIQUE command seems to take an inordinate amount of time and you suspect that it is runaway, press **Esc** to terminate the INDEX command operation and ERASE the incomplete .NDX and .W44 files.

TOTAL

If you TOTAL ON <key field> TO <filename> with a database file that has no numeric fields, you may get an "Internal error: EVAL work area overflow" or "Internal error: Illegal opcode" error message. The resultant file is created and control returns to the dot prompt. Occasionally, this type of TOTAL operation returns the error message "Internal error: EVAL work area overflow," the DOS error message "Divide overflow," and then the computer freezes, requiring a cold boot.

TRANSFORM

Using TRANSFORM() with an argument combining the @(and @B functions displays extraneous characters following the output. For example,

```
? TRANSFORM(-10,"@(B9999")
(10.00          )) <---- Result.
```

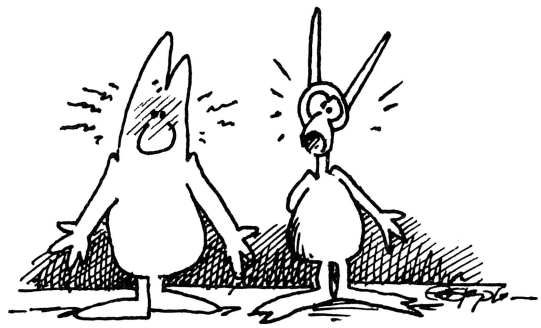
The extra characters may be another parenthesis, a field name, or the argument from a previous function.

Framework II

@writetextfile/@writeframefile

There are two problems when using long frame names with **@writetextfile** and **@writeframefile**.

1. There is a display anomaly if a **@writetextfile** or **@writeframefile** is executed multiple times without closing the file cabinet. This results in multiple entries of the file name in the cabinet. Closing the file cabinet and reopening it clears up the problem.
2. If a frame name contains from 1 to 96 characters, the entry in the DOS directory will have the correct eight-character name and the proper extension. If the frame name contains 97 or 98 characters, the extension will be missing one character for each character past 96. 97 characters will produce a two-character extension and 98 characters will produce a one-character extension. If the frame label contains 99 or more characters, the extension will be dropped completely.



HI-80 Plotter

When a graph containing a value on the X-axis is plotted using the Epson HI-80 plotter driver, one of two things may occur.

1. Framework II drops back to DOS with a "Divide overflow" error message. When this happens you must reboot your computer, or when you reload Framework II, the computer may freeze.
2. The message "Frame not drawn" may be displayed at the bottom of the screen and Framework II continues without error.

Telecommunications/UPL

Use of the local command **UPL** (UPLoad) command in telecommunications to upload (.FW2) files, freezes the computer, requiring a warm boot. The work-around is to use the **Send as Text** option from the Apps:Telecommunications:Text Transfer menu or **COPY (F8)** to upload files.

VT-100 Terminal Emulation

Some mainframe systems send three-digit escape sequences for cursor positioning. Framework II, in VT-100 emulation, only accepts a two-digit escape sequence. In the following two examples, the number "005" is not interpreted correctly by Framework II. The "05" in the second example works, however.

Host computer sends:
{esc}(005f
└——— 3 digits for cursor positioning.

Host computer sends:
{esc}(05f
└——— 2 digits for cursor positioning.

There is no work-around.

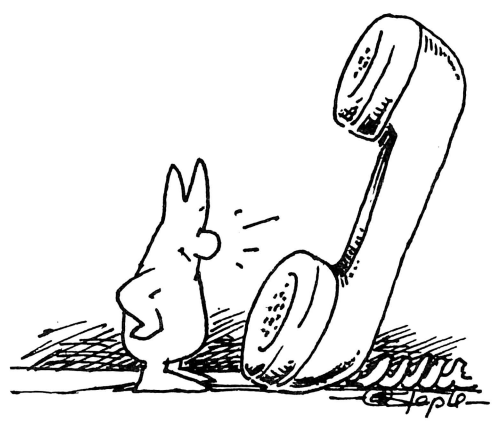
ON-LINE SERVICES

by Kent Irwin

Do you have a modem? Do you subscribe to CompuServe or The Source? If so, we have great news for you! Ashton-Tate offers technical support on all of these electronic information services. You can quickly access the kind of quality technical information that you have come to expect from ***TechNotes***, and you can do it through your computer system. To take advantage of this progressive new support method, you must subscribe to one of the services listed below. There are no additional charges for using our on-line support services, apart from the normal connect-time fees for each service. You can get specific subscription information by calling one of the following numbers:

Service	Subscriber Information	Command to find Ashton-Tate
CompuServe	(800) 848-8199	GO ASHTON
The Source	(800) 336-3366	ATSIG

To give you an idea of the fast-moving dynamic support that is happening on-line, ***TechNotes*** will be publishing regular excerpts from the Ashton-Tate Forum on CompuServe. This monthly material comes from messages our Forum users have posted on the bulletin board. To retain the original sense of this material, we have edited only for spelling, and all message headers have been removed to protect the identities of our users. Every other message within a single conversation topic has been italicized to enhance readability.



dBASE II, dBASE III, dBASE III PLUS

dBASE III PLUS is quite compatible with the AST RAMpage! and DESQview on an XT. Matter of fact, it will run in the background or in a window (as many as you have memory for). I'm not doing anything special with DESQview, but I can give you my setup if you need it. As far as the RAMpage! goes, there is no need to do anything special.

I run dBASE III PLUS on an AT&T 6300 with an AST Six-Pak and an AST RAMpage! board (2 MB). I have had no problem installing or running. Ashton-Tate does specifically support the RAMpage! with Framework II, and they are not aware of any difficulty with dBASE III PLUS (though it doesn't use the extra memory).

One thing about dBASE III PLUS and DESQview is that PLUS will run in a small window without problems and can be run in multiple windows at the same time. One of our installations that we brag most often about involves two complex dBASE III applications; one of which is about 90% compute-bound and the other is about 90% operator dependent. The first is run in the background window most of the time doing reports and batch updates while data entry is done in the foreground window. It saved the customer from buying a second PC-AT, so they're happy, and it's a very impressive system for us to demo.

That sounds like an interesting system. I may have to dust off my DESQview and give it a try. Though I will admit that Windows has captured my multiprocess interests for the moment.

It's quite an example for illustrating what a computer can do with the right hardware and software. If I was selling dBASE, I'd certainly let folks know about it. What can you do with Windows, other than play Reversi? I don't think that there is any software out yet that fully supports it; thus no multi-tasking, only paging to disk. But it may be the system of the future.

.....

A few days ago you suggested that I could send a report that I sent to disk while in dBASE II by using the DOS command of COPY < filename.ext > LPT1:. First, thank you for answering. Second, do you know of any way, while still in dBASE II to accomplish this end? I would prefer to have the operator have all of her commands within the program itself, and it is my understanding that you cannot execute DOS commands while in dBASE II. Your advice would be appreciated.

dBASE II does not have this capability. About the best you could do is to use the QUIT TO command to branch out of dBASE II to do the output. Then, in that QUIT TO command, include a call to return to dBASE II.

.....

One of our corporate dBASE users is experiencing a rather odd problem. He is using an application that takes him three menu levels into the application and issues a command that paints a new screen while using a .DBF with a relation set. (Does this all make sense ?) In any event, the problem is he is getting an error to the effect of "Disk Full while writing file: Abort, Retry, Ignore:" The odd part is, at this particular part of the program, no file writing should be going on. Rather, if anything, the program should be performing a read to verify the relation. The disk full error seems additionally erroneous in that neither abort, retry, or ignore solves the problem. No matter what he does, the entire system gets locked and the system must be rebooted to continue.

Try hitting Esc right before the problem occurs and then SET ECHO ON and SET STEP ON. Then you will see the lines in the program as they execute, and be prompted to hit Spacebar to continue after each one. This way you will know exactly what statement triggers the problem. Just a hunch, but I think you may have a spelling error or something that triggers weird stuff (has happened to me).

If the message really said "Abort, Retry, Ignore?" then that's a DOS error, not a dBASE error. In all cases I have seen, the messages that come up "Abort, Ignore, Delete files?" are dBASE errors, and the other is from DOS. Double check the message and make sure which it is.

Is your client using DOS 3.0? One of its lovely little bugs is popping up that error message for no apparent reason. If so, changing to DOS 3.1 solved the problem(s) for me every time.

.....

I was recently backing up my database when I noticed a file with the same name as my database but with a .TBK extension. What is this file? I have been using dBASE III and have not seen anything like it before. I should note that the database works fine.

The .TBK file is a backup of the database's memo file. When you perform an operation like MODIFY STRUCTURE, the memo file .DBT extension is backed up to a file with the .TBK extension. The .TBK file can be deleted, with no ill effects.

.....

Help! When I use WordStar as my MODIFY COMMAND editor, the .PRG file it generates looks fine on the screen, but when run produces a syntax error and shows the program line written in the odd IBM characters that look quite strange. Works ok when I restore the dBASE III PLUS editor, break up the file into small parts, load it, save it, and put it all back together again. This is not fun. Solution?

You are using WordStar in the D (document) mode - or else someone has been peeking at your code with WordStar in the D mode. Source code must be written with an ASCII text editor. WordStar in the N (nondocument) mode fills the bill. But you should reinstall WordStar to come up in the N mode by using the WordStar install program.

Anyone know how to embed compressed print control codes into a standard REPORT FORM print command from the keyboard ? Would appreciate help! Know it must be simple, my dilemmas always are!

You have to initialize your printer before starting the report. From the keyboard, enter:

```
. SET PRINT ON
. ? CHR(15)
. SET PRINT OFF
```

Then type your REPORT FORM command. The example is for Epson compressed print. Substitute whatever code your printer requires.

.....

Hi, can anybody answer this question for me?

I'm trying to write a dBASE program for work to keep track of all our tools and programs (I work in a machine shop). I'm using a total of four databases to help speed the LOCATE and FIND commands. My question is can I input data into one of the databases and have the program input it into the other databases? I'm using similar fields in all four databases.

I've read the manual on using the JOIN command but I don't really understand how to go about doing it. I'm not a programmer, I'm a machinist that likes to tinker. Can anyone explain how to do what I want to do in simple terms?

Take a close look at the UPDATE command and see if it won't do what you want. In addition, if you're using like data in all four files, you might consider going to one large file even if some of the fields are blank for some records. Ease of manipulation and speed should be improved.

.....

I have a Symphony database that I want to convert to dBASE III PLUS. I use the TRANS.COM program supplied by Lotus to convert to a dBASE II format, then use the dBASE dCONVERT to convert the dBASE II file to dBASE III PLUS. However, I am running into a problem where the field names come out all right but the actual database is not lined up properly with the fields. Hence, I would find the phone number (or part of it in the name field); part of an address in the other fields. How can I rectify this problem?

Can Symphony create an SDF or ASCII type file, or one delimited with commas? In that case, you could write it out from Symphony and use APPEND FROM <filename> to read it in to dBASE. If the fields don't match you can adjust the structure of the dBASE file accordingly.

You may want to try converting to a Lotus spreadsheet first; then use the APPEND FROM TYPE WKS or whatever the syntax is. Make sure you follow the rules in the dBASE III PLUS manual, mainly removing column labels first. Also, dBASE III PLUS will import DIF files, SDF, or delimited.

Framework II

How can I alter the Rundbase.FW2 FRED program to look for my DBASE.COM file in another directory on C: (\DBASE), rather than on A:? I'm not a programmer, but it doesn't seem as if it would be too hard to do. Another question: Do I really want to do this? I have a 400K + dBASE file I need to access, and only 640K tops in my machine. Does the program throw enough of Framework II out of memory to make room for a file this size?

Try this:

```
; Rundbase
@eraseprompt,@local(frame),@echo(#ON),
@if(@memavail < 256000,
@list(@eraseprompt,@beep,
@prompt("Not enough memory to run dBASE III -- Press a key to
continue"),
@nextkey,@eraseprompt,@return(#FALSE)
)
),
@prompt("Loading..."),
@setdirectory("c:\DBASE"),
@run(frame,"DBASE.COM")
```

The size of your database file is not important. You will need a minimum of 256K available for dBASE III, and I recommend 320-384K if you are using dBASE III PLUS. None of the memory used by the Framework II main program is swapped out. Only the desktop (data) space is available as free memory in DOS Access. Keep in mind that you can use your dBASE database files in Framework without using dBASE (assuming you want to examine or analyze data, rather than add data to the file).

The @setdirectory did the trick! I even added another @setdirectory to the end of the program to put me back in touch with my FW files. My God! I'm programming! Sort of...

.....

I want to configure Framework II to save files automatically to a subdirectory called \FW\FRAMES. I have a DOS batch file that includes the instruction CD \FW\FRAMES, but this doesn't appear to be enough. The frustrating thing is that I had it working fine with Framework 1.1, but haven't be able to duplicate my success with Framework II. I know I must be forgetting something obvious, but what?

P.S. The new EGADCL.SC driver works like a charm! The character set is a little basic, but much easier on the eyes. And those graphs!

A piece of cake. Have your batch file start Framework II like this:

FW Start.FW2/x

Then, create and save a frame called Start that has this formula:

@setdirectory("C:\FW\FRAMES")

You will automatically start up with the "frames" directory as your default.

.....

A couple of questions/problems with Framework II I'd like to set up a field formula to force a text field into upper case. The @Upper user-defined function described in a recent ***TechNotes*** does the conversion, except... The field formula looks like this:

< text field > := @UPPER(< text field >)

But when text is entered into the field, the function returns with error, #VALUE! appears in the field. It works fine if the text is entered into the formula portion of the field as a string value (interestingly, the returned value appears in the text portion of the field). Why can't the text portion of a database field be passed to a function as a parameter in this way? It will work if the field is "calculated" from the text in another field.

I don't think there's a way to successfully do that. I would use a separate field, or a formula separate from the database entirely. I have written a Framework II "Picture" function, designed to simulate the essence of the dBASE function. However, it is a user-defined function, and is designed to be used in a formula which is controlling data entry.

.....

MULTIMATE TECHNICAL SUPPORT

Sharing Data with MultiMate Advantage

by Liz Burke and Karen Robinson

Since the merger of Ashton-Tate and MultiMate International, many people have asked how to use dBASE III PLUS and Framework II files with MultiMate Advantage. This article discusses how to import these files into a MultiMate primary merge document.

To begin, the MultiMate mailmerge option accepts several types of files: sequential files (delimited files), random files (SDF files), MultiMate list documents, and MultiMate Info-Handler datafiles. Here we will focus on how to merge sequential or delimited files with MultiMate, since both dBASE III PLUS and Framework II can export to them.

Creating a Delimited File

The delimited files that MultiMate accepts can have a maximum of 64 fields per record. Fields are separated by commas and are optionally delimited with quotes. For information on how to specify a different field delimiter, refer to the section "Merging with a Database" in the *MultiMate Advantage Advanced User's Guide*.

In dBASE III PLUS, memo fields are the only field types not supported by the DELIMITED option. Date fields are copied as numeric in the format YYYYMMDD. Logical fields are copied in the following format: T for true and F for false. Numeric fields are copied in the format defined by the field, and character fields are delimited with quotes. For example, if your file has the four field types, character, date, numeric, and logical, the delimited file looks like this:

"Liz",19860912,123.00,T

To format the fields in a particular way, such as spelling out dates, you will have to write a custom program to create the delimited file. The sample program below creates a delimited file that copies a date field written out in character format, and spells out the logical values. To use the program, replace Yourfile with your database filename, and the following field names with the field names from your database file: Character, Date, and logical.

```
SET TALK OFF
USE Yourfile
SET ALTERNATE TO Temp.TXT
SET ALTERNATE ON
DO WHILE .NOT. EOF()
   * ---Send character field delimited with quotes.
   ?? '"'+ Character field + '"'+ ","
   * ---Send date field out as character, spelled out.
   ?? '"'+ CMONTH(Date field);
      + " " + LTRIM(STR(DAY(<Date field>) ,2))
      + ", " + STR(YEAR(<Date field>),4)
```

```
+ '"'+ ","
* ---Send numeric field out as is.
?? <<Numeric field<>
* ---Write out the value of the logical field.
?? IIF(<Logical field>, "," + '"'+ "True" + '"'+ ;
   '"'+ "False" + '"')
* ---Record separator.
?
SKIP
ENDDO
SET ALTERNATE OFF
SET ALTERNATE TO
```

Using the data from the example above, the delimited file looks like this:

"Liz","September 12, 1986",123.00,"True"

For more information on creating custom formats for delimited files, refer to the Usage Tip, COPY TO [DELIMITED] in the May 1986 issue of ***TechNotes***.

To create a standard delimited file in dBASE III PLUS, USE the database file, then execute the COPY command:

COPY TO <text filename> DELIMITED

If you want to specify which fields to include in the delimited text file, use the FIELDS option of the COPY command. The FIELDS clause must be used when copying from a View. If it is not used, only the fields from the current database file are copied. If you want to limit the number of records copied, either filter the database using SET FILTER or use the FOR clause or scope parameter within the COPY command. For example, if you want to COPY just the name and address fields of your database file, the COPY TO command might look like this:

```
COPY TO Clients FIELDS Name, Address, ;
        City, State, Zip DELIMITED
```

To create a delimited file in Framework II, highlight the frame border of the database or spreadsheet, then select the **dBASE Delimited** option from the Disk:Export submenu.

Importing a Delimited File into MultiMate

The MultiMate primary merge document has two parts: the define block, which identifies the fields in the delimited file, and the letter itself, which contains the text of the letter and the field names to merge.

Once you have created the delimited file, follow the steps below to create a MultiMate form letter to merge data from your delimited file.

1. If you are using the MultiMate pre-menu, select option 1, **MultiMate Professional Word Processor** to get to the main menu. At the MultiMate main menu, select item 2, **Create a New Document**.
2. At the "Create a New Document" screen, enter the name of the form letter that you are going to create. Press **Return** to go to the "Document Summary" screen.
3. Press **F10** twice to move beyond the "Document Summary" and the "Modify Document Defaults" screens. This brings you to the first working page of your form letter.

As stated earlier, when merging data into a letter, MultiMate requires you to define (give names to) the fields in the delimited text file. Preceding the text of the letter is the define block of the document. It begins with the word "Define" delimited with **Alt-M**, and ends with "End Define," also delimited with **Alt-M**. When you press **Alt-M**, the symbol "╞" displays on the screen. Below is an example of a MultiMate merge document.

```
╞Define╡
File Type Sequential
Field Name Name
Field Name Address
Field Name City
Field Name State
Field Name Zip
╞End Define╡

Dear ╞Name╡

Thank you for your letter regarding...
```

4. To begin the define block, move the cursor to line 1, column 1, and type the following:

   Alt-M Define**Alt-M Return**

5. On line 2 type:

   File Type Sequential **Return**

   This tells MultiMate the type of text file to merge with, sequential or delimited.

6. Next, you need to identify each field in the delimited file by giving each a name. The field names are on separate lines, in the same order shown in the delimited file, preceded by the phrase Field Name. It is imperative that every field in the sequential text file be identified in the define block in the correct order. The easiest way is to use the same field names used in the database file. Referring to the example above, the five fields copied to the delimited text file might look like this:

   Field Name Name
   Field Name Address
   Field Name City
   Field Name State
   Field Name Zip

7. To end the define block, type the following on the line after the last field:

   Alt-M End Define**Alt-M Return**

8. The define block is now complete. What follows is the text of the letter. Type the letter and embed the field names within the text. All field names must be capitalized as they are in the define block, and delimited with **Alt-M**. The following example shows how to place City, State, and Zip on the same line:

   Alt-M City**Alt-M**, **Alt-M**State**Alt-M Alt-M**Zip**Alt-M**

MultiMate offers several special merge commands that allow you to repeat text, print records more than once, and suppress blank lines. For details, refer to the "Merge" section of the *MultiMate Advantage Reference Manual.*

9. Once the letter is complete, press **Alt-5**. This saves the letter taking you directly to the "Merge Print a Document" screen.

10. Press **Return** to select **Merge Document Name**. The default is the name of the form letter you just created.

11. Now the cursor should be highlighting the option **Merge Data File**. If it is not, move the cursor there and enter in the name of the sequential or delimited file, including the extension.

12. Press **F10** to advance to the "Print Parameters for Document" screen. This is where you set up left and right margins, print pitch, etc. For information on how to use this menu, refer to the section "Print Document Utility of Screens and Menus" in the *MultiMate Advantage Beginner's Guide*.

13. To begin printing the letters, press **F10**. The prompt on the screen should say "Please wait...printing a document." If for any reason you need to stop the printing process, press **Esc** twice to cancel the print request.

Now you should have a general idea of how to merge data from dBASE III PLUS and Framework II with MultiMate Advantage. For specific information on the format of dBASE III PLUS delimited files, refer to the COPY or APPEND commands in the Commands section of *Learning and Using dBASE III PLUS*, as well as the article "Sharing Data with Other Programs" in the June 1985 issue of ***TechNotes***. For more information on using MultiMate's mailmerge capabilities, refer to "Merging with a Database" in the *MultiMate Advantage Advanced User's Guide*.

How to Sort and Print Info-Handler Datafiles

by Charles Gagliardi

The Information Handling feature in MultiMate Advantage is designed to help you more easily create and control data for use with the MultiMate Merge utility. With Info-Handler datafiles, you can search and replace text within a field for a specific condition. These datafiles can be merged with form letters, mailing labels, and preprinted forms. You can also print just the datafile.

One feature of Information Handling that you may find particularly useful is its sorting capability. Using this feature, you can sort, in ascending or descending order, any list of information you wish. Some examples are mailing and membership lists. The example below demonstrates how to create, sort, and print a mailing list.

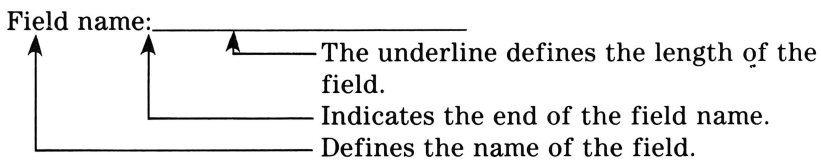
Follow these steps to accomplish this task:

1. Design the template (datafile structure) for data entry.
2. Enter the data, one name and address per record.
3. Sort the data in ascending or descending order.
4. Create and print the merge document.

Creating and Sorting Information Handling Datafiles

To create and add records to the data entry template, follow these steps:

1. From the MultiMate main menu, select option 2, **Create a New Document**.
2. At the "Create a New Document" screen, enter the datafile name (e.g., Maillist).
3. Enter the information for the "Document Summary" screen, and then press **Shift-F10** to move to the first page of the Info-Handler template.
4. This is where you create the template design (datafile structure). Each field name has three elements: a field name, a colon, and an underline wide enough for your longest entry. The generic format of each field is as follows:



A typical template for names and addresses looks like this:

Name: _____
Address: _____
City/State: _____
Zip: _____

Enter the field names in the format shown. If you want, you can place more than one field per line.

5. When you are finished designing the format, press **F2** to save it.
6. Answer Y to the prompt, "Are you sure you want to leave the template page? (Y/N)." This takes you to the first record of the data entry screen.
7. The data entry template you just created is displayed. Enter the data, and press **Return** as each field entry is complete.
8. After the last field is entered, press **F2** to move to a new blank record. Do not press **F2** after entering the last record.
9. After the last record is entered, move the cursor to the field you want to sort on, such as Lastname, then press **F5**.

The sort order in Information Handling is defined by the ASCII codes. Numbers are placed before letters and uppercase letters appear before lowercase. To sort numerically, you must use leading zeros (e.g., 1 should be entered as 001).

10. The upper right side of the screen says "Sort Order: 1 ascending, 2 descending, 3 abort." Press either 1 or 2. The screen prompt then says "Please wait..."
11. When the sort is complete, the first record in the datafile displays. To scroll through the records, use **Ctrl-PgUp** and **Ctrl-PgDn**.
12. Press **F10** to save the datafile and return to the MultiMate main menu.

The Information Handling datafile is now created and in sorted order.

Printing

As stated before, there are many ways to print this data. You can list it or merge it into letters, preprinted forms, or mailing labels. You can also select a range of records to print. For a record to print, it must be SELECTED. The choice of SELECTED or UNSELECTED is made during data entry and editing. To print a partial list of the records, change the unwanted records to UNSELECTED. There are two ways to do this:

1. Press **Alt-F1** to UNSELECT/SELECT the current record.
2. Press **Alt-Y** to UNSELECT the current record and all remaining records.

For more information on SELECTing and UNSELECTing records, refer to the "Information Handling Data file Create/Edit" section in the *MultiMate Advantage Reference Manual.*

To print an alphabetized list of the datafile, follow these steps:

1. From the MultiMate main menu, select 2, **Create a New Document.**
2. At the "Create a New Document" screen, enter the document name (e.g., List).
3. Next, enter the information for the "Document Summary" screen, then press **F10** to advance to the first page of the merge document.

4. Type the field names in the format that you want them to print. You can include some or all of the fields from the Info-Handler datafile. The field names must be capitalized exactly as they are in the data entry template and delimited with **Alt-M**. When you press **Alt-M**, you will see the "C" on the screen. If you want MultiMate to suppress blank lines due to empty fields, use the option OB, delimited with **Alt-M** after the field name. For example:

   Alt-M Name**Alt-M**
   Alt-M Address**Alt-M**Alt-MOB**Alt-M** **Alt-M**City/State**Alt-M**
   Alt-M Zip**Alt-M**

   results in

```
├Name┤
├Address┤├OB┤ <---------If the Address is blank, do
├City/State┤  ├Zip┤     not print a blank line
```

   The example above prints one record per page. If you want more than one per page, use the MultiMate Repeat command. The Repeat command is also delimited with **Alt-M**. The following example prints 10 names and addresses per page. The blank line preceding End Repeat ensures that a blank line prints between each record. For example,

   Alt-Mrepeat:10**Alt-M**
   Alt-Mnext**Alt-M**
   Alt-M Name**Alt-M**
   Alt-M Address**Alt-M**
   Alt-M City/State**Alt-M** **Alt-M**Zip**Alt-M**
   Alt-Mend repeat**Alt-M**

   results in

```
├repeat:10┤
├next┤
├Name┤
├Address┤
├City/State┤ ├Zip┤

├end repeat┤
```

5. When you finish designing the printing template, press **F10** to save the document and return to the main menu.
6. From the MultiMate main menu, select 5 **Merge Print Utility**.
7. At the "Merge Print a Document" screen, enter the name of the merge document (List) and the merge datafile (Maillist).
8. Press **F10**, and then select a range of records to print.
9. Press **F10** again to move to the "Print Parameters for Document" screen, and enter the print parameters.
10. Press **F10** to print the list.

From this article you have learned how to create, sort, and print an Information Handling datafile. For information on other ways to use these datafiles, such as merging the data files with form letters, refer to the section "Information Handling" in the *MultiMate Advantage Reference Manual.*

INTERNATIONAL SUPPORT

by Paul Bukauskas

Available Language Versions

The popularity of dBASE, Framework, and MultiMate has resulted in numerous foreign language translations. If you are a developer and would like to pursue vertical markets, you will find a listing of these translations useful. Additional translations are constantly being added to this list. We will inform you as new translations become available.

International Translations

Version	Product/Version					
	dBASE III		Framework	Framework II	MultiMate	
	1.1	PLUS	1.1	1.0	3.30	3.31
Australian/ New Zealand	Yes[a]	Yes[a]	Yes[b]	Yes	—	Yes[b]
British	Yes[a]	Yes[a]	Yes	Yes	—	Yes
Danish	Yes	—	Yes	—	Yes	—
Dutch	Yes	—	Yes	—	—	Yes
Finnish	Yes	—	Yes	—	—	—
French	Yes	Yes	Yes	Yes	Yes	—
German	Yes	Yes	—	Yes	—	—
Italian	Yes	Yes	Yes	Yes	—	—
Japanese	Yes[c]	—	—	—	—	—
Norwegian	Yes	—	Yes	—	Yes	—
Portuguese	Yes	—	Yes	—	—	—
Spanish	Yes	—	Yes	—	—	Yes
Swedish	Yes	—	Yes	—	Yes	—

[a] U.S. version of dBASE III PLUS.
[b] British version of Framework.
[c] The Japanese version is 3.0J.

Usage Tips for International Users

MODIFY COMMAND

Memo (.DBT) and program (.PRG) files created using the U.S. versions of dBASE III and dBASE III PLUS are not compatible with the European versions of dBASE III. MODIFY COMMAND in the U.S. version uses "soft" carriage returns for word wrapping; this consists of a carriage return (0A) and linefeed (0D) with the high-order bit set. All of the European versions use these characters as part of the printable character set. So, if you attempt to edit a memo field or .PRG file that contains "soft" carriage returns, you will see only the first 80 characters of the file and the rest are truncated. You can work around this by replacing the "soft" carriage returns (0A 0D) with "hard" carriage returns (00 FF) prior to editing these files in a European version of dBASE III. This can be done either by using an external text editor that inserts "hard" carriage returns at the end of each line or by using SCON, a utility program included with every European version of dBASE III.

Index Files

Index files created in the European versions of dBASE III are not always usable in the U.S. versions of dBASE III and dBASE III PLUS. This is because the European versions of dBASE III support the extended character set (ASCII 128–255) and the U.S. version does not. In essence, if your index key contains any ASCII value above 127, the resultant index file is not usable in the U.S. versions of dBASE III or dBASE III PLUS.

LETTERS

Hiding Character and Numeric Data

Dear ***TechNotes***:

Champion Accounting Software hides data in database fields by masking the fields with various character graphics. Can you explain how this is done?

J.C.

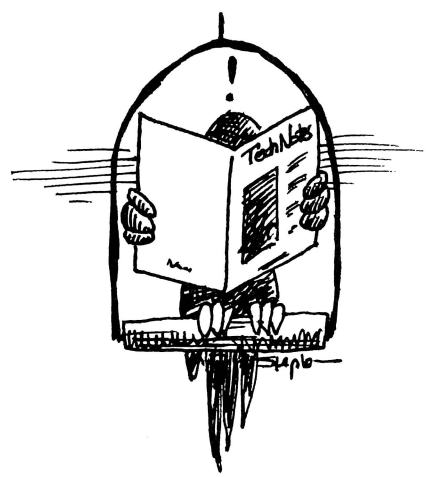
Dear J.C.:

Sometime management wants to protect data or make it unavailable to others. A good example is an employee file that contains different salaries. Obviously, management does not want this information to be public knowledge; so, there must be some way of hiding this information. All data files in dBASE II and dBASE III are stored as ASCII characters; therefore, anyone can view this information. In dBASE III PLUS, there is a utility called PROTECT that will encrypt database files. However, dBASE II and dBASE III users don't have access to this utility. For them, it becomes a little tricky.

As you may know, every character on the keyboard is represented by a numeric (ASCII) code. We can make use of ASCII codes to encrypt and unencrypt data. Here's how this works. Although the ASCII codes range between 0 and 255, inclusive, the majority of the characters we know can be represented by the ASCII codes between 0 and 127, inclusive. For example, the letter "A" has an ASCII code of 65. dBASE II version 2.4 and greater includes the RANK() function, which you can use to obtain the ASCII code of any character. In dBASE III, you can use the ASC() function. For example:

```
? RANK("A") <--------dBASE II
? ASC("A")  <--------dBASE III
```

returns 65 in both cases.



Conversely, to find the character representation of the ASCII code, use the CHR() function in both dBASE II and dBASE III. From our previous example:

```
? CHR(65)
```

returns "A."

Therefore, we use the CHR() function in conjunction with the ASC(), or RANK() function, to encrypt and unencrypt information. First, use the ASC() function to get the ASCII code, subtract that result from 255 (the highest ASCII code), then use the CHR() function to turn it back into a new character. The main drawback here is that this must be done for each character in a string. Also, someone may be able to figure out the encryption if the same method is used for every character.

Let's go back to our original example of encrypting salary information and assume that Names.DBF contains a field called Salary. Usually this field is a numeric field, but for encryption purposes, we need to make it a character field with a width of 10 characters. This should not pose a problem for pay increases, because we can use the VAL() function to find the numeric value of a character string, make the necessary changes, and use the STR() function to change it back to a character string.

For example, suppose employee A has a salary of $10,000. Since Salary is a character field with a width of 10, this is represented as " 10000". The first character is a space and has an ASCII code of 32. Using the scheme described above, subtract 32 from 255 (which results in 223) and use the CHR() function to obtain the new character. The complete formula for the first character is:

```
STORE CHR( 255 - ASC( SUBSTR( Salary,1,1) ) ) TO mv1
```

which results in "_".

Use the same code to unencrypt the character.

To stop someone from figuring out the encryption scheme, you may want to change the 255 in the example above to 219 for the second character. Mix them up. Just be sure that the same number is used to unencrypt the file. Once each character has been encrypted, use the REPLACE command to put the contents into that field. That is:

```
REPLACE Salary WITH mv1 + mv2 + mv3 +... + mv10
```

SETting Proper RELATIONs

Dear ***TechNotes***:

I am using dBASE III and have two database files. Vendor.DBF contains the vendor name, address, and code. Vendor.DBF is INDEXed ON Code TO Vcode. Trans.DBF contains transactions and includes a vendor code. Trans.DBF is INDEXed ON Code TO Tcode. There are no deleted records in either database file.

By setting up the database files as follows, I am able to view fields from both files.

```
SELECT 2
USE Vendor INDEX Vcode
SELECT 1
USE Trans INDEX Tcode
SET RELATION TO Code INTO Vendor
```

However, if I set up the files as follows:

```
SELECT 2
USE Vendor INDEX Vcode
SELECT 1
USE Trans INDEX Tcode
SELECT 2
SET RELATION TO Code INTO Trans
```

I am unable to retrieve information from Trans.DBF. Why doesn't it work this way?

J.B.

Dear J.B.:

In your first example, when you SET RELATION TO Code INTO Vendor, you are currently in the Trans work area. Internally, dBASE III takes the Code field from Trans.DBF, moves to Vendor.DBF and performs a SEEK to position the record pointer at the first occurrence of the Code field from Trans.DBF. If the SEEK fails to locate the Code from Trans.DBF, then the record pointer moves to the end of the file, thus giving blank information. dBASE III then returns back to the Trans work area.

Assume Code in Vendor.DBF is six characters in length and Code in Trans.DBF is eight characters in length. Therefore, you are able to SEEK Vendor->Code in Trans, because the length of Code in Vendor is less than the length of Code and Trans. However, if you try to SEEK Trans->Code in Vendor, no information will be retrieved, because dBASE III cannot find an eight-character code in a six-character field. This is the only reason why the RELATION will work in one direction and not the other, assuming that some information is common in both Code fields.

Framework II Printing Problems

Dear ***TechNotes***:

As a Framework user, I naturally upgraded to Framework II when it became available. I have both an Epson LQ-1500 and a Qume printer. One problem I noticed with Framework II that I did not have with Framework is that double quotes appear at the beginning of printed text. This only occurred on the Epson LQ-1500 printer and I thought it was a bad printer driver. Later, I discovered that when I used the **Starting Control Codes** option in the Print menu to configure the printer for various features, I separated the various code strings with commas. Of course, it was these commas that printed. You might consider pointing out this potential error in ***TechNotes*** as it is a natural mistake for a person with FRED programming experience to make.

L.J.L.

Dear L.J.L.:

Thanks for the information.

Light Pen Interface with dBASE III PLUS

Dear ***TechNotes***:

Do you know of any hardware or software package that can interface a light pen to dBASE III PLUS? I have looked around and have found none thus far.

O.B.

Dear O.B.:

TechNotes cannot recommend any software or hardware not made by Ashton-Tate. dBASE III PLUS has a LOAD and CALL command. These commands allow a user to LOAD a .BIN file into memory and execute it with the CALL command. If you know how to program in assembler, you can write a program to implement this. If this is beyond your range of programming, try to find an assembler programmer. If this is too expensive, check with a local computer users group. If you want information on a dBASE users group nearest to you, contact Marcy Glickman here at Ashton-Tate, (213) 329-8000.

Suppressing Zeros Using REPORT FORM in dBASE II and dBASE III

Dear ***TechNotes***:

I read in an earlier issue of ***TechNotes*** how to use the IIF() function in dBASE III PLUS for suppressing zeros in a report form. This is a great feature, but I am using dBASE II. Is there any way of suppressing zeros in a REPORT FORM in dBASE II?

M.J.

Dear M.J.:

Yes, there is a way to suppress zeros in a REPORT FORM. However, like dBASE III PLUS, these columns cannot be totaled. This formula will also work for those users who have dBASE III and have not updated to dBASE III PLUS.

Assume that Num is a numeric field with a width of five. In the dBASE II REPORT FORM, put the following command line in the field contents:

```
5 , $ ("      0" + "      " + STR(Num,5) + STR(Num,5), ;
@(STR(Num,5),"      0      " + STR(Num,5) + STR(Num,5) ) + 5,5)
```

In the dBASE III REPORT FORM, place this command line in the field contents:

```
SUBSTR("      0" + "      " + STR(Num,5) + STR(Num,5), ;
AT(STR(Num,5),"      0      " + STR(Num,5) + STR(Num,5) ) + 5,5)
```

First, a 20-character string is broken into four groups of five characters. The first five characters is the character string representation of 0 (i.e., STR(0,5,0)). The second five-character group is spaces. The third and fourth five-character group is the character representation of the field Num. Using the @ function in dBASE II (AT() function in dBASE III), locate the position where STR(Num,5) appears in the 20-character string. If Num is 0, the @ function returns a 1 (the first character in the first group). Otherwise, the @ function returns an 11 (the first character in the third group). Next, we move five positions (to the next group of five characters) and output the next five characters. If Num is 0, we output five spaces (the second group of five characters). Otherwise, we output the number (the fourth group of five characters).

READER SUBMISSION FORM

If you have a programming tip, a handy formula, or a short utility, and you are interested in sharing it with other Ashton-Tate or MultiMate enthusiasts in a future issue of ***TechNotes***, please read the provisions below. After reading these guidelines, fill out this form and send it along with your submission(s) to the following address. Submissions must be made on disk (IBM or compatible format) in standard ASCII format or in Framework II (.FW2) format.

***TechNotes* Submissions**
Software Support Center
Ashton-Tate
20101 Hamilton Ave.
Torrance, CA 90502-1319

NAME _____ DATE _____

ADDRESS _____

CITY _____ STATE _____ ZIP _____

PHONE (During normal business hours) (_____) _____ - _____

_____

(SIGN HERE TO ACKNOWLEDGE YOUR UNDERSTANDING OF THE PROVISIONS BELOW)

To establish a clear understanding of the rights and usage of submissions, you are asked to first represent and warrant that:

1. You are the author of the material: that it is original and has not been heretofore published; that it does not infringe upon any copyright, proprietary right or any other right whatsoever; that it contains no matter that is contrary to law; and that you are the sole owner of the rights herein conveyed to us, and that such rights are free and clear.
2. We shall have the right to use your name, biography or other information pertinent to your submission.
3. We shall have the right, at our discretion, to edit the material to suit the requirements of our publication.
4. Suggestions, comments, and programming materials submitted to ***TechNotes*** are not for reply. Materials submitted for publication are considered contributions to the "pool" of information found in ***TechNotes*** for the advantage of our readers. Although, we do not pay or in any other way compensate you for your contribution, your name shall appear as author or contributor.

Thank you for reading ***TechNotes***. If you do not have a submission for us, write us anyway and let us know what you think about our publication. We certainly appreciate any feedback you may have.

Ashton-Tate Compatibility Guide

The Ashton-Tate Compatibility Guide is now available!

The Compatibility Guide is an information-filled packet describing third-party products that increase the application possibilities of Ashton-Tate software products.

The packet concentrates on sophisticated micro-to-mainframe links and expanded memory boards. Products covered include the Above Board from Intel (which allows much larger spreadsheet applications to be developed) and Software AG's Natural/ Connection (which allows data to be exchanged between PCs and mainframe computers).

To obtain a copy of the Compatibility Guide, please fill out the form below, enclose a check for $3 (to cover postage and handling) made payable to Ashton-Tate, and send to:

Ron Arons
Marketing Programs Manager
Ashton-Tate
20101 Hamilton Avenue
Torrance, CA 90502-1319

Name _____

Company _____

Address _____

City _____ State _____ ZIP _____

TN

TechNotes Documentation Survey

Ashton-Tate wants to provide you with product documentation that meets your learning needs. The following questions are designed to help us gather information about the types of documentation Framework II users find most helpful. Your answers will be used to assist our Documentation department in improving the effectiveness of our product manuals.

The first part of the questionnaire requests information about you and your computer system, because documentation that would be most useful to one user may not be most useful to another. The second part of the questionnaire requests information about your reactions to the Framework II documentation. When you have completed the questionnaire, use the enclosed postage-paid envelope to return it to us. Thank you for your time and feedback.

1. What is your personal computer experience?
   ___ Novice personal computer user
   ___ Intermediate personal computer user
   ___ Advanced personal computer user, non-programmer
   ___ Advanced personal computer user, programmer

2. On what type of computer system are you using the product:
   a. Drives:
   ___ PC with 2 floppy disk drives
   ___ PC with a hard disk drive
   ___ Part of a network of PCs
   b. Monitor: ___ Monochrome ___ Color
   c. Internal memory (RAM):
   ___ 128K ___ 256K ___ 384K ___ 512K ___ 640K
   ___ more than 640K
   d. Hard disk storage capacity:
   ___ 10M ___ 20M ___ 30M to 40M ___ more than 40M

3. Did you notice that there was an on-disk tutorial? ___ Yes ___ No

4. Did you use the on-disk tutorial to learn Framework II? ___ Yes ___ No

5. Was the on-disk tutorial adequate to get you started? ___ Yes ___ No

6. Did you use the printed tutorial, *Learning Framework II*, to learn the program?
   ___ Yes ___ No

7. Is it a good idea to expand the Help system to include more of the documentation?
   ___ Yes ___ No

8. Which is your one, most preferred way to learn Framework II?
   ___ Printed manual ___ Videotape
   ___ On-disk tutorial ___ Laserdisk
   ___ Audiotape ___ Classroom instruction
   ___ Work with user group
   ___ Looking at someone else's frames, work, or FRED source code
   ___ Other _____

9. How often do you use the HELP feature in the Framework II software?
   ___ Often ___ Sometimes ___ Rarely ___ Never

10. How useful do you find the HELP feature in the Framework II software?
    ___ Excellent ___ Good ___ Neutral ___ Fair ___ Poor

11. What one thing would most improve the Help feature?

_____

12. How do you rate the Framework II documentation?

	Excellent	Good	Neutral	Fair	Poor
Ease of learning	___	___	___	___	___
Clarity of information	___	___	___	___	___
Sufficient information	___	___	___	___	___
Usefulness of examples	___	___	___	___	___
Quantity of examples	___	___	___	___	___
Organization of information	___	___	___	___	___
Usefulness of illustrations	___	___	___	___	___
Attractiveness of manuals	___	___	___	___	___

13. How often do you use the following parts of the Framework II documentation:

	Often	Sometimes	Rarely	Never
Getting Started	___	___	___	___
Learning Framework II	___	___	___	___
Using Framework II	___	___	___	___
Framework II Advanced Topics	___	___	___	___

14. Did you know that Framework II has its own programming language, called FRED? ___ Yes ___ No

15. How many FRED programs have you developed?
___ None ___ 1-10 ___ 11-25 ___ 26-50 ___ more than 50

16. How useful was the documentation in teaching you to build FRED language programs? ___ Excellent ___ Good ___ Neutral ___ Fair ___ Poor

17. What one thing would improve the documentation's presentation of the FRED programming language?

_____

18. Do you use Framework II for telecommunications? ___ Yes ___ No

19. How useful was the documentation in teaching you to use Framework's telecommunications features?
___ Excellent ___ Good ___ Neutral ___ Fair ___ Poor

20. What one thing would improve the documentation's presentation of Framework II telecommunications?

_____

21. What is your single, most important Framework II learning need that is not currently being met?

_____

22. Does your organization provide training in how to use Framework II?
___ Yes ___ No

23. If your organization provides Framework II training, what learning needs were met that were not satisfied by the documentation?

_____

24. Are you interested in participating in further research into the learning needs of Framework users? ___ Yes ___ No

If yes, please provide your name, address, and phone number below.

_____

_____

_____

SUBSCRIBE TODAY TO *TechNotes*!

Don't miss a single issue of Ashton-Tate's hottest source of technical know-how! Each monthly issue of ***TechNotes*** provides you with the most current, accurate technical information available from the Software Support Center on all Ashton-Tate and MultiMate products.

Call our toll free number to subscribe:

(800) 331-4164 (In California)
(800) 828-2514 (Residents of other states)
(619) 487-3212 (In Canada)

If you prefer, fill out the form below and return it with payment to:

***TechNotes* Subscription Service**
P.O. Box 3729
Escondido, CA 92025-0929

— — — — — — — — — — — **(CUT HERE)** — — — — — — — — — — —

NOW ACCEPTING CREDIT ORDERS! Call our toll free number for details. Make checks payable to ASHTON-TATE.

Amount Enclosed:

__ $50.00

__ $65.00 (Extra Postage Canada and Mexico)

__ $75.00 (Extra Postage other foreign countries)

__ Bill Me

Form of Payment: (check one)

__ Check (**U.S. dollars drawn on U.S. bank**)

__ VISA Number _____

__ MasterCard Number _____

Expiration Date _____

Signature _____

Check One: __ New Subscription __ Renewal

Name _____

Company _____

Address _____

City _____ **State/Province** _____

Zip/Postal Code _____ **Country** _____

Which product(s) do you use? __ **dBASE II** (D2)
__ **dBASE III** (D3)
__ **Framework** (FW)
__ **MultiMate** (MM)

Please allow 30 to 60 days for delivery of your first issue. 1000

TechNotes
BACK ISSUES ORDER FORM

Maybe your subscription to ***TechNotes*** just began and you want to have every issue. Maybe you don't subscribe, but want to receive last month's issue. Maybe you want to refer to an article in a past issue to help you design that special application.

Whatever the reason, order Back Issues! Add to your library of dBASE II, dBASE III PLUS , and Framework II technical reference material. To order your Back Issue(s), fill out this form, enclose your payment, and return to:

***TechNotes* Back Issues**
Ashton-Tate Software Support Center
Attn: Kathy Smith
20101 Hamilton Ave.
Torrance, CA 90502-1319

Name _____

Company _____

Address _____

City _____ State/Province _____ Zip Code _____

Country _____ Phone Number (_____) _____-

All 1985 Back Issues are $10.00 each, when you order them separately. If you purchase all available 1985 issues as a set, the price is $50.00. All 1986 Back Issues are $7.00 each. All prices include postage and tax.

The following issues are currently available. Check those you wish to order:

( ) FEBRUARY 1985	( ) SEPTEMBER 1985	( ) JANUARY 1986
( ) MARCH 1985	( ) OCTOBER 1985	( ) FEBRUARY 1986
( ) APRIL 1985	( ) NOVEMBER 1985	( ) MARCH 1986
( ) MAY 1985	( ) DECEMBER 1985	( ) APRIL 1986
( ) JUNE 1985	(includes 1985	( ) MAY 1986
( ) JULY 1985	Annual Disk and	( ) JUNE 1986
( ) AUGUST 1985	1985 Cumulative	( ) JULY 1986
	Index)	( ) AUGUST 1986

$_____ TOTAL AMOUNT ENCLOSED

( ) VISA ( ) MasterCard # _____

Expiration Date _____ Signature _____

ALL ORDERS MUST BE PREPAID. MAKE CHECKS PAYABLE TO ASHTON-TATE (IN U.S. DOLLARS DRAWN ON U.S. BANK). PLEASE ALLOW SIX (6) WEEKS FOR DELIVERY OF YOUR ORDER.

NOTE: Information contained in ***TechNotes*** is current as of the issue in which it appears. Therefore, some information found in the Back Issues may no longer be valid (such as Update Policies, etc.).

Notes

Notes

Notes

Notes

TechNotes
P.O. Box 3729
Escondido, CA 92025-0929

ADDRESS CORRECTION REQUESTED

BULK RATE
U.S. POSTAGE
PAID
PERMIT NO. 27
CULVER CITY, CA